(1)

AFIT/GOR/ENS/88D-25

IMPROVING THE SURVIVABILITY OF A STOCHASTIC COMMUNICATION NETWORK

THESIS

Eugene Yim
Captain, USAF

AFIT/GOR/ENS/88D-25

Approved for public release; distribution unlimited

AFIT/GOR/ENS/88D-25

IMPROVING THE SURVIVABILITY OF A STOCHASTIC COMMUNICATION NETWORK

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
In Partial Fulfillment of the
Requirements for the Degree of
Master of Science in Operations Research

Eugene Yim, B.S. and M.E.A.
Captain, USAF

December, 1988

Approved for public release; distribution unlimited

Preface

This research investigated an analytical approach to measure the performance of a stochastic communication network and to determine the investment (improvement) strategy that maximizes network performance. This thesis contains the results of six months of research to accomplish this goal. The methodology developed in this research can be applied to any stochastic network.

The successful completion of this research would not have been possible without the support of many outstanding individuals. I am deeply thankful to my thesis advisor, Dr. Yupo Chan, whose professional guidance was invaluable throughout this research. I have gained a tremendous learning experience from his research advice and suggestions. I am also very thankful to my thesis reader, Major Kenneth W. Bauer, for his valuable suggestions and great optimism throughout this research effort. Especially, his famous morale-boosting smile was inspirational. I owe many thanks to my friends, Captain Eddy Clark and Captain Tom Glenn Bailey. Eddy's advice on programming in Prolog was very helpful during the developing stage of the program, FORMULA. Glenn, who has worked on the same research project using the simulation approach, has brought the spirit of team work through close coorporation. I would like to express my gratitude to my thesis sponsors for providing such an interesting and challenging research project on which I have enjoyed working. Finally, I thank my parents who constantly supported and encouraged me throughout the time I was assigned to AFIT. This research is dedicated to them.

Eugene Yim

Table of Contents

List of Figures

List of Tables

Abstract

This research examined the performance of communication networks with stochastically failing components; it also investigated investment strategies to improve network performance. The performance of the network was measured in terms of the amount of information that could be handled. Since the exact calculation of expected maximum flow, which measures the performance of a stochastic network, is mathematically intractable, the bounds of expected maximum flow were investigated. The network performance, both under normal and adverse conditions, was measured analytically by formulating the problem as a linear programming model. The improvement of network performance was made by increasing the capability of the components to handle the information. Mixed integer programming models were developed to determine the investment strategy which maximized the lower bound of expected maximum flow.

Because of the complexity involved in developing these models for a large network, a Prolog program was written to generate formulations for all performance and investment strategy models based on the arc-path incidence matrix. Such formulations serve as a direct input file to off-the-shelf mathematical programming packages such as the LP/MIP-83 linear and mixed integer programming system. In this research, three "realistic" communication networks of various sizes and topologies were analyzed. The solutions so obtained provided the performance of the network, identified bottleneck components in the system, identified significant paths in the system, and generated optimal investment strategy to improve the network performance. The methodology employed in this research would be useful in analyzing other stochastic networks.

IMPROVING THE SURVIVABILITY OF A STOCHASTIC COMMUNICATION NETWORK

I. Introduction

This chapter contains a general backgound of research related to the analysis of communication networks whose components are subject to failure. The specific purpose of this research is stated with the listing of the research objectives. Also included are the scope of the research, the assumptions considered throughout the research, and the equipment used in this research.

1.1 Background

Communication systems can be modeled as a network flow model. In this model, the transmitter, processing sites, and receiver, which are geographically dispersed, can be represented as nodes; the communication channels can be represented as arcs through which nodes are connected.

The arcs and nodes of the network are susceptible to failure when they are influenced by severe weather, conventional war, and other adverse conditions. Such threats make arcs and nodes *stochastic*; that is, the survivability of these components are probabilistic. The stochastic nodes and arcs are characterized by the maximum amount of information (capacity) they can handle and the probability of survival during adverse situations. Under normal conditions, when the network is perfectly reliable, the performance of the network can be measured by solving the maximum flow through the network. Under adverse conditions, when the network is under some kind of threat such as missile attack, the performance can be assessed by measuring the expected maximum flow through the network using predetermined survival probabilities and capacities of components.

Unfortunately, the problem of finding the exact value of the expected maximum flow is infeasible for a large stochastic network. The amount of computation involved to solve the problem increases exponentially as the number of the stochastic components increases. For a network with n stochastic arcs subject to only two possible states, 2^n separate topologies (failure states) have to be enumerated to compute the probability for each topology. As an example, a network containing 55 stochastic arcs has 2^{55} or approximately 3.6028797 x 10^{16} failure states. A computer with the processing time of 100 nanoseconds (10^{-7} seconds) – which is almost twice as fast as the 16 MHz IBM-AT computer – per state would take approximately 114.25 years to enumerate all possible states! Thus, finding the exact solution for a large network is infeasible in real time. An alternative method to assess the stochastic network performance is to measure computationally feasible lower and upper bounds of the expected maximum flow.

1.2 Research Problem

The purpose of this research is to examine the performance of communication networks whose components are subject to failure and to determine the investment (improvement) strategy that maximizes network performance.

1.2.1 Research Objectives Directed toward the accomplishment of the above research goal, the following specific research objectives have been generated:

1. Assess the network performance under normal conditions.
2. Assess the expected network performance under adverse conditions.
3. Determine the investment strategies to improve the expected performance of a stochastic network.
4. Solve formulated problems by using mathematical programming packages, such as LP/MIP 83 [31], and analyze the results.

1.3 Scope

This research focused on measuring the expected system performance of the stochastic network and finding the investment strategy to improve the expected system performance. There are two possible approaches to this research problem: a simulation approach or an analytical approach. This research concentrated on finding the solution analytically by formulating a system using mathematical programming methodology.

The network performance, both under normal and adverse conditions, was measured by formulating the network problem as a linear programming model. Both linear and mixed integer programming models were developed to determine the investment strategy. Emphasis was given to the formulations of the lower and upper bounds of the expected maximum flow and investment strategy. The formulated problems were solved using an off-the-shelf mathematical programming package, LP/MIP 83. The solutions so obtained identified bottlenecks in the system, provided performance of the network, and generated an optimal investment strategy to improve the performance of the network.

1.4 Assumptions

The following assumptions were made throughout the research, unless otherwise stated:

- Arcs and nodes are subject to total failure; the components can only be in one of two possible states: *up* or *down* with a certain probability.
- The network does not contain any cycles.
- Arcs are one-way arcs; the flow is permitted in one direction only.

1.5 Equipment

The equipment used in this research were computers and software. A majority of the work was done on IBM-AT compatible microcomputers. The Arity/Prolog Version 5.0 [2, 3] was used to write the Prolog program to generate all mathematical models required to do the research. An Arity/Prolog and various mathematical programming packages, including the LP/MIP-83, were installed on the microcomputers. These computers have the following specifications:

- 512 kilobytes of random access memory (RAM)
- 20 megabyte hard disk
- 8 MHz motherboard
- one or two double-sided double-density $5\frac{1}{4}$ inch floppy disk drives
- 8087/80287 numeric co-processor (required to run LP/MIP-83 with .Y87 version; .N87 version does not require an 8087/80287)

If the reader is interested generally in knowing what things to consider in using the computer for operations research, refer to specific references [6, 14].

II. Literature Review

This chapter presents the review of literature applicable to the analysis of network performance. The notation used in this thesis is described in the first section. Network representation and maximum flow formulation are reviewed. The formulations for calculating expected maximum flow and bounds of expected maximum flow of a stochastic network are also presented. Following each formulation, the solution algorithms are briefly discussed. Finally, a basic concept of Prolog programming is introduced.

2.1 Notation

The following notations are used in this thesis. Although, they are explained in the text as they appear, they are presented here for easy reference.

α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

a_{ij} = An element of arc-path incidence matrix: 1 if arc i lies on path j; 0 otherwise

B = Total budget available for investment

b_i = Predetermined amount of capacity increase in component i

c_i = Cost of increasing the capacity of component i by one unit

d_i = Decision variable (continuous) denoting how much capacity to increase in component i

$e(u_i)$ = Expected capacity of component i

e_{ik} = An element of arc-node incidence matrix: -1 if arc i starts at node k; 1 if arc i ends at node k; 0 otherwise

f_j = Flow on path j

g_i = Decision variable (integer) denoting the number of investments that can be made in component i.

n = Number of arcs in the network

p_i = Probability of survival of component i

P_k = Probability that the failure state is k

q = Number of paths in the network

r = Any designated node; source, intermediate, or sink node

R_j = Reliability of path j

s = Source node

s_k = Failure state k; describes the capacities of the components when the network is in state k

t = Sink node

u_i = Capacity of component i

V = Feasible flow through the network from s to t

V_k = Flow through the network from s to t when the failure state is k

x_i = Flow on arc i

y_i = Maximum amount of capacity increase allowed in component i

2.2 Network Representation

A network is a collection of arcs and nodes. Each arc and node contain information, such as flow and capacity, that characterizes the network. A network representation is useful for modeling a wide range of physical situations, such as water, transportation, and communication systems. *Flow* in the network usually models some flow quantity in the network such as the flow of water, cars, or information. In many situations, the arcs that connect two nodes are *directed* in a sense

that the flow on the arc is permitted in one direction only. An example of such arcs is a one-way street, where traffic can only move in one direction.

The flows on the arcs are controllable within constraints set by arc capacities, conservation of flow, and node external flows. *Capacity* is the maximum amount of flow that an arc and node can handle. It is the parameter defining the upper bound for flow. The minimal amount of flow that arcs and nodes must handle is refered to as *lower bound.* Normally the lower bound is zero. *External flow* is the required quantities of flow entering or leaving the network at each node. The law of *conservation of flow* states that the flow entering the node from the arcs plus the external flow at the node equals the flow leaving the node on the arcs of the network.

The structure of the network can be described fully in a graphic form or in a matrix form. A graphic form, such as the one shown in Figure 1, is most commonly used. Two common ways of representing the network structure in a matrix form are an arc-node incidence matrix and arc-path incidence matrix. In an arc-node incidence matrix, the rows correspond to arcs and the columns correspond to nodes. The elements of the arc-node incidence matrix, e_{ik}, are defined as -1 if arc i starts at node k, as 1 if arc i ends at node k, and 0 otherwise. In an arc-path incidence matrix, the rows correspond to arcs the same as before, but the columns represent paths in the network from source to sink. The elements of the arc-path incidence matrix, a_{ij}, are defined as 1 if arc i lies on path j, and as 0 otherwise.

2.3 Maximum Flow

The maximum flow problem is "the problem of finding a maximum flow through a prescribed network" [13:370]. A sample deterministic network is shown in Figure 1. The node s is the source, the node t is the sink, and the nodes 2 and 3 are the intermediate nodes. The arcs are one-way arcs in which flow is allowed in one direction only. Each arc has flow, x_i, and flow capacity, u_i, where i is the arc number. The amount of flow from node s to t is denoted by V. Assuming there are no external

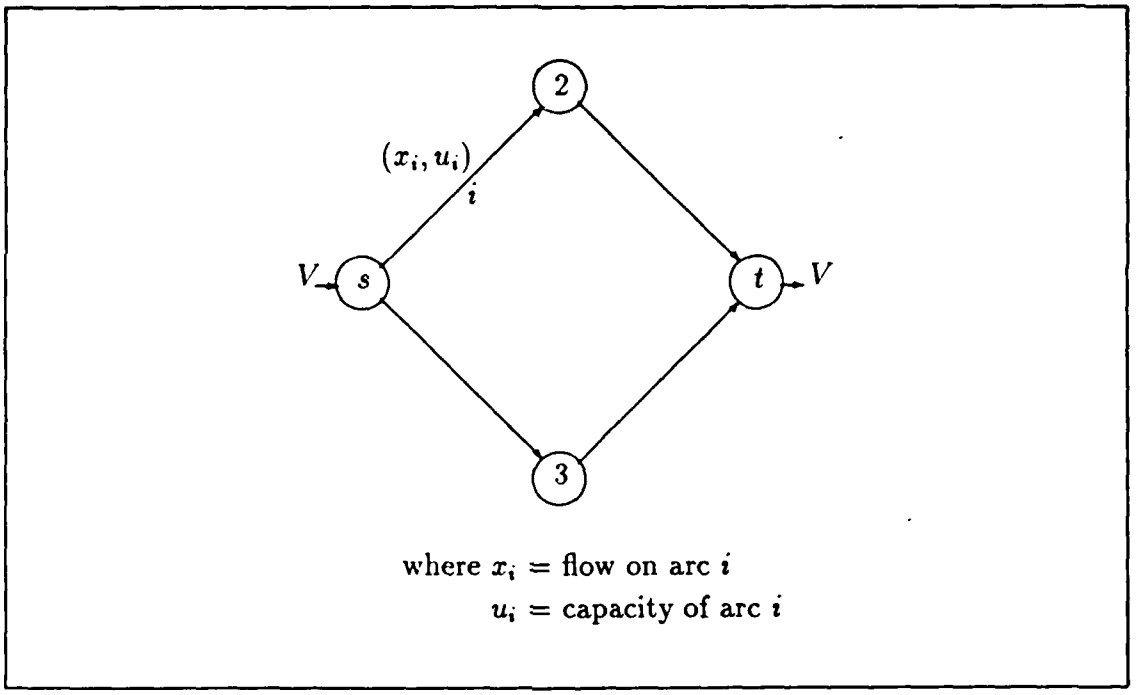


Figure 1. A Sample Deterministic Network

flows at the nodes, the flow into a node equals the flow out of a node at intermediate nodes, that is, at intermediate node r,

$$\sum_{i \in \alpha} x_i = \sum_{i \in \alpha'} x_i$$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r,

and the flow on each arc is constrained by capacity, that is,

$$0 \leq x_i \leq u_i$$

.

2.3.1 Formulation Specifically, the maximum flow problem is finding the "maximum flow that can be sent out of a source node s into a terminal node (sink) t, such that the flow in each branch is within the capacity, and that at each node

other than (s,t), the flow is conserved, that is, $\sum flow\ in = \sum flow\ out$" [23:7]. The mathematical formulation of the maximum flow problem is shown in Table 1.

Table 1. Formulation of Maximum Flow

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$0 \le x_i \le u_i \qquad i = 1,2,...,n$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

Solving the maximum flow problem not only gives maximum flow, but also detects the bottlenecks in the network. At optimality, if an arc has flow that is equal to the capacity of that arc ($x_i = u_i$), then that arc is the bottleneck. To make improvements, the capacities of the arcs that form the bottlenecks can be increased, but the improved network may again contain bottlenecks without reaching the optimal solution. Hence, a better approach is to formulate a problem to find how much to invest in arc capacities so that the maximum flow, V, becomes as large as possible within the available budget, B.

Let c_i be the cost for increasing the capacity of arc i by one unit, and d_i be the decision variable that denotes amount of capacity increase in arc i. Then, the problem can be formulated as shown in Table 2 [12, 33].

Table 2. Formulation of Maximum Flow With Improvement Strategy

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$\sum_{i=1}^{n} c_i d_i \leq B$

$0 \leq x_i \leq u_i + d_i \qquad i = 1, 2, ..., n$

$d_i \geq 0$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

If any of the nodes in the network shown in Figure 1 also have a capacity, the problem can still be formulated as before by replacing a node with "two nodes joined by one dummy arc," [24:15] as shown in Figure 2. A dummy arc, representing the capacitated node, is constrained by the nodal capacity.

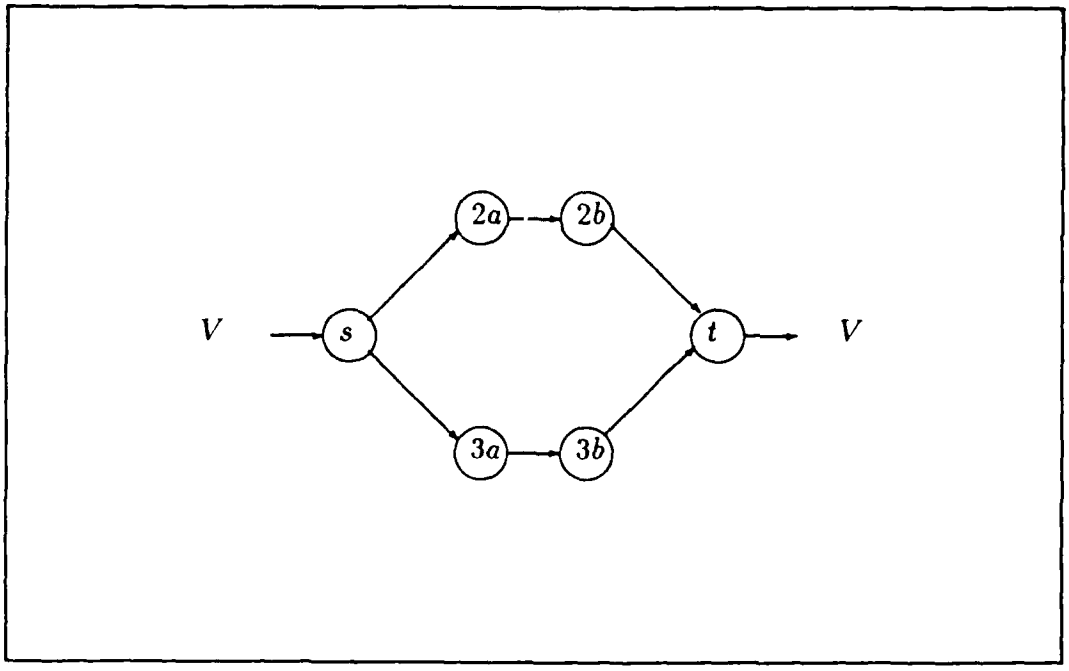


Figure 2. Converted Network with Arc Capacities Only

If there are more than one source and sink in the network, where flow can go from any source to any sink, the problem can be converted to the simple single source and node maximum flow problem. This can be accomplished by adding a new artificial source s and sink t, with added arcs leading from s to every source node and from every sink to t, as shown in Figure 3 [24:15]. The added arcs have unbounded capacity.

2.3.2 Discussion of Solution Algorithms Ford and Fulkerson [17] developed a solution algorithm based on a max-flow min-cut theorem. The theorem states that "for any network with a single source and sink, the maximum feasible flow from source to sink equals the minimum cut value for all of the cuts of the network" [19]. Their algorithm is called a *labelling algorithm*, and it has been the basis for other researchers, such as Edmonds and Karp [15], for developing a more efficient algorithm. According to Lin, et al. [23], Edmonds and Karp's algorithm, called *two-end labelling*, typically saves about 20% of computing time compared with the labelling

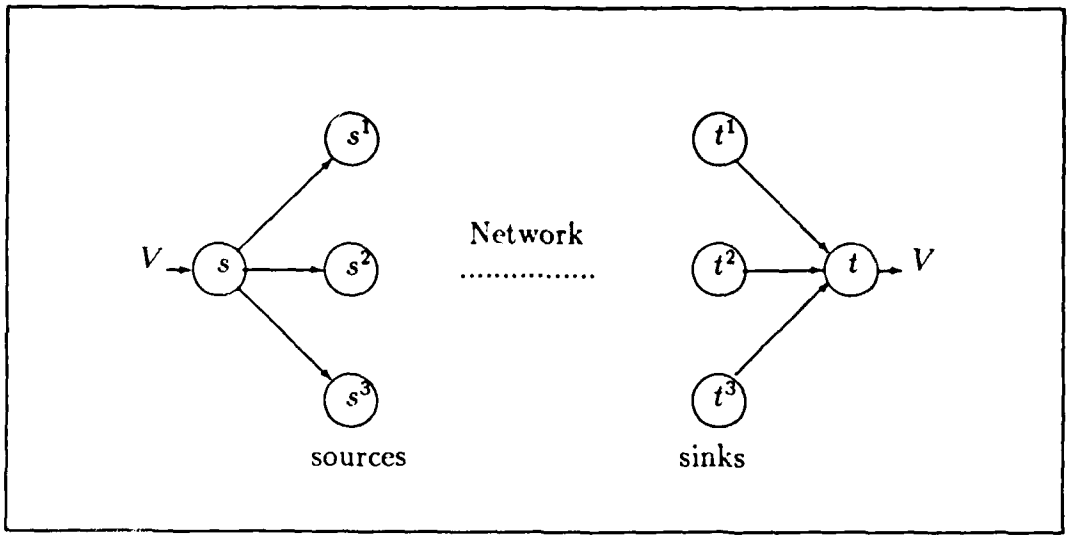


Figure 3. Converted Network with Single Source s and Sink t

algorithm of Ford and Fulkerson. A detailed discussion on the new algorithm and a copy of the computer program is included in the report by Lin, et al.

2.4 Expected Maximum Flow

A network containing arcs or nodes, collectively referred to as components, that are subject to failure is referred to as *stochastic network.* These components, under adverse conditions, are subject to *total* or *partial* failure [33:87]. The components that are subject to total failure can be in one of two possible states: *up* or *down* with a certain probability. If the components can have capacities between these two extremes, then the components are subject to partial failure. The components are characterized by survival probabilities and capacities. For a stochastic network, the network performance can be measured by computing the expected value of the maximum flow from source to sink with known survival probabilities and capacities.

To find the exact value of the expected maximum flow, all possible network topologies (failure states) have to be generated to calculate the probability for each failure state. A simple stochastic network is shown in Figure 4.

The network in Figure 4 contains three arcs, each with a survival probability

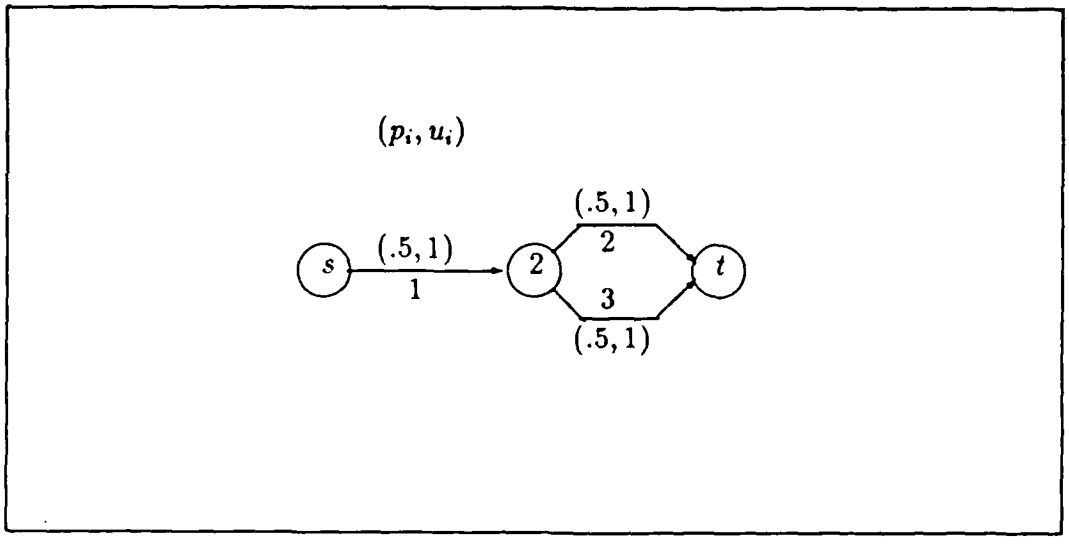


Figure 4. A Stochastic Network

of .5 and a capacity of 1. The number of possible failure states, assuming arcs are subject to total failure, is $2^3 = 8$, as shown in Figure 5. Assuming the independence of component failures, the probability of occurrence of each state can be obtained by applying a simple probability rule. This can be obtained by multiplying the survival probabilities, p_is, for existing arcs and failure probabilities, $(1-p_i)$s, for non-existing arcs. As an example, the probability of occurrence of the second failure state, shown in Figure 5, is $P_2 = (.5)(.5)(1-.5) = .125$ with a maximum flow of one. For failure states 4 through 8, there is no path from source to sink, thus maximum flow for those states is zero. The expected maximum flow is $(.125)(1) + (.125)(1) + (.125)(1) = .375$.

2.4.1 Formulation The general formulation, implementing the above idea, to find the exact value of the expected maximum flow is shown in Table 3.

2.4.2 Discussion of Solution Algorithms The problem of finding the exact value of the expected maximum flow is classified as NP-hard which simply means that the computational effort grows exponentially with the number of stochastic components. Currently, there does not seem to be any efficient solution algorithm

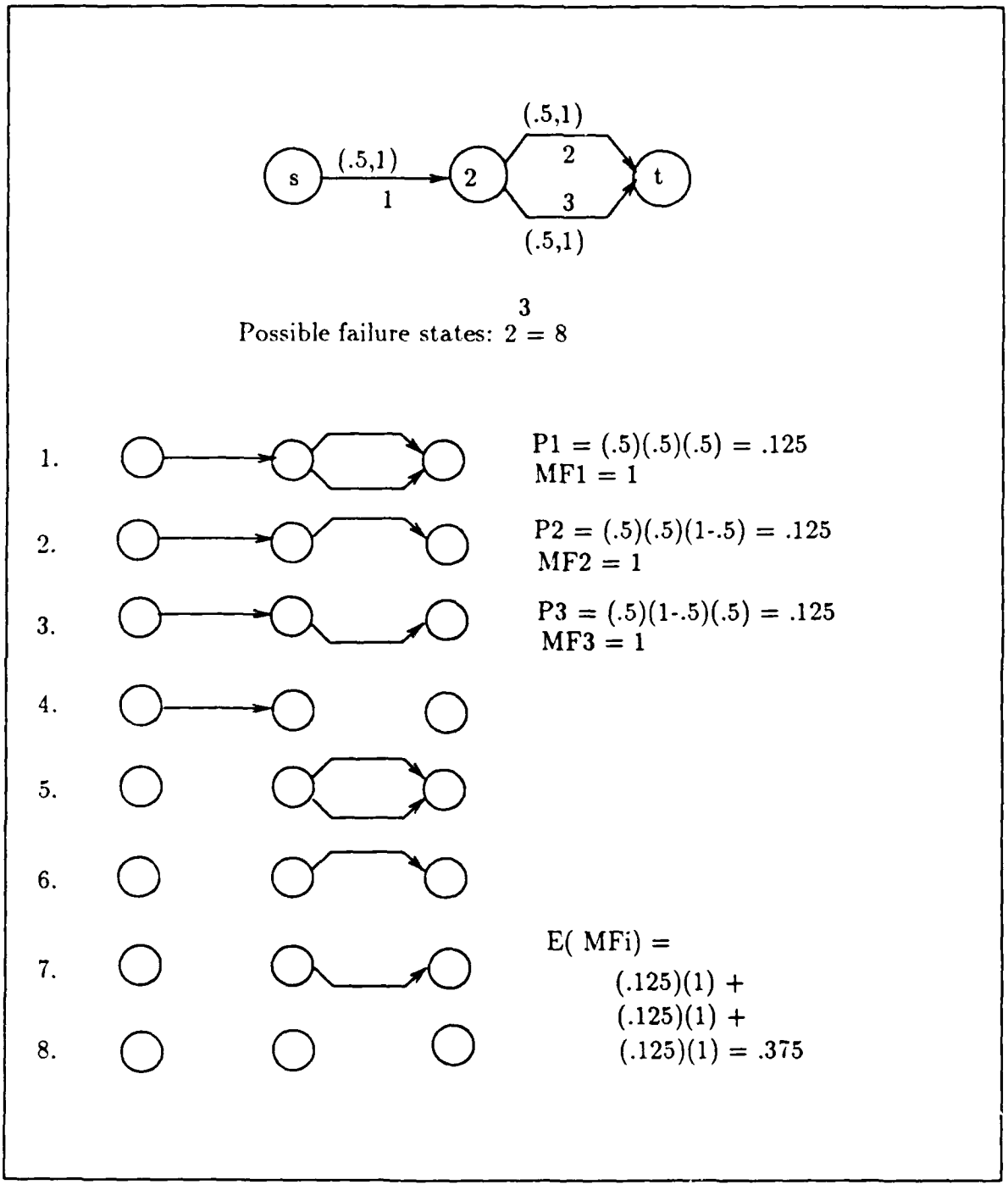


Figure 5. Eight Failure States of the Network Shown in Figure 4

Table 3. Formulation of Expected Maximum Flow

$$\sum_{k=1}^{2^n} F(s_k)P_k$$

where s_k = Failure state k; describes capacities of the arcs when the network is in stake k

P_k = Probability that the failure state is k

$F(s_k)$ is

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$0 \leq x_i \leq u_i \qquad i = 1, 2, ..., n$

where V_k = Feasible flow through the network when it represents failure state k

available for finding the exact value of expected maximum flow. Evans [16] developed a solution methodology which involved enumeration of all possible cuts. Aneja and Nair [1] viewed Evans' method as not very useful from a computational viewpoint. Wallace has simplified Evans' method somewhat, but he still views the method to be computationally infeasible [33]. However, an algorithm [30] has been recently developed which enables the calculation of the approximate value of expected maximum flow by "enumerating 'most probable' states" [30:516] of the network. This algorithm is an improved version of the algorithm proposed by Lam and Li [22].

2.5 Lower Bound of Expected Maximum Flow

As previously mentioned, finding the exact value of the expected maximum flow becomes infeasible as the number of the stochastic components in the network increases; thus, finding the bounds, instead, is suggested to assess the network performance. Although bounds do not provide a true single-value expected maximum flow, they do provide valuable information. In many situations, especially when designing a new system, it may be more important to know the worst expected performance rather than just knowing the expected performance of the network. The lower bound provides this pessimistic appraisal of the stochastic network performance.

In a stochastic network, when components fail, the path(s) that contain(s) failed components become inoperative and the flow along that path(s) reduce(s) to zero. To ensure the flow is maximal for the new state of the network, the flows along functioning paths may have to be altered or rerouted. The expected maximum flow formulation, which was already discussed in the previous section, implicitly assumes rerouting takes place when components fail. However, when rerouting is not possible for some reason, the flow on that failed path(s) reduce(s) to zero and the flows along functioning paths cannot be altered. In this case, the expected network performance or expected maximum flow subject to inability to rerouting is the lower bound for the expected maximum flow. The lower bound is sometimes referred to as *maximum*

expected flow [1]. The maximum expected flow (lower bound) can be obtained by maximizing the sum of the expected capacities of the paths from source to sink, since the expected flow through the network is equal to the sum of the expected capacities of the paths from source to sink [29].

2.5.1 Formulation Consider any stochastic network. Let a_{ij} $(i = 1,2,...,n;$ $j = 1,2,...,q)$ denote the elements of the arc-path incidence matrix where

$$a_{ij} = \begin{cases} 1 & \text{if arc i lies on path j} \\ 0 & \text{otherwise} \end{cases}$$

Assuming all nodes are 100% reliable, and each arc i in the network has a survival probability of p_i $(0 < p_i \leq 1)$, the reliability of the path j, R_j, is the product of the survival probabilities of the arcs on that path:

$$R_j = \prod_{\{i|a_{ij}=1\}} p_i$$

$$\text{for } j = 1,2,...,q$$

Let f_j be the flow on path j. The sum of the expected path flows from source to sink can be stated as $\sum_{j=1}^{q} R_j f_j$. The lower bound, the expected maximum flow subject to inability to reroute, is given by the formulation shown in Table 4.

The objective is to maximize the sum of expected path flows from source to sink while path flows are constrained by arc capacities. Note that when R_j is 1 for all j, this formulation provides the maximum flow through the network. This maximum flow formulation is based on the arc-path incidence matrix whereas the one described earlier is based on the arc-node incidence matrix.

2.5.2 Discussion of Solution Algorithms Carey and Hendrickson employed Aneja and Nair's algorithm, which employs a simplex method, to find the maximum expected flow. Recently, Sancho [29] has come up with his own algorithm which

Table 4. Formulation of Lower Bound

objective function

$$\text{Max} \sum_{j=1}^{q} R_j f_j$$

subject to

$$\sum_{j=1}^{q} a_{ij} f_j \leq u_i \qquad i = 1, 2, ..., n$$

$$f_j \geq 0 \qquad j = 1, 2, ..., q$$

is based on the maximum flow problem developed by Ford and Fulkerson. Both of these formulations require finding all the paths and calculating the reliabilities. In this research, a Prolog program was written to generate the paths and their reliabilities.

2.6 Upper Bound of Expected Maximum Flow

Knowing the upper bound of the expected maximum flow is perhaps not as critical as knowing the lower bound to assess the system performance. However, the upper bound provides an optimistic appraisal of the network performance which may be useful in some application.

2.6.1 Formulation Onaga [26] has shown that "the expected value of maximum flow is less than or equal to the flow on a network with [arc] capacities set equal to the expected [arc] capacities" [10:444]. Using Onaga's finding, the upper bound can be obtained by formulating a problem the same as the maxmum flow problem, except that the arc capacity is now set to the expected arc capacity, $e(u_i) = p_i u_i$.

The upper bound of expected maximum flow can be formulated as shown in Table 5.

Table 5. Formulation of Upper Bound

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$0 \leq x_i \leq p_i u_i \qquad i = 1, 2, ..., n$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

2.6.2 Discussion of Solution Algorithms The discussion of the solution algorithms of upper bound formulation is the same as the one for the maximum flow problem.

2.7 Overview of Prolog

This section discusses the basics of Prolog computer language; Prolog was used to write a program to generate mathematical programming models needed to do the

research. This section is intended for the reader who is not familiar with Prolog.

Prolog stands for "*PROgramming in LOGic*" [7:80]. It was developed in the early 1970s and was originally designed to "assist in the analysis and comprehension of natural language" [7:80]. Over the years, it gradually gained popularity and became a prominent language, along with LISP [9], in the area of artificial intelligence [28].

As its name implies, Prolog uses logic as a programming language. It changes the way of writing computer programs for solving problems [7]. In high-level language, such as Fortran, Pascal, and Ada, the instructions are written in the program to tell the machine (computer) specifically what to do. In a logic programming environment, however,

> the user no longer has to think in terms of commands to be given to a machine, but only in terms of describing the nature of the problem, leaving it to the machine to translate the problem specification into a set of commands for execution [7:81].

2.7.1 Characteristics of Prolog A Prolog program consists of *facts* and *rules*. Facts represent information about relevant relationships between objects. They declare things that are always true, such as "man is a human". Rules, on the other hand, specify things that may be true if some conditions are satisfied. For example, "X is the mother of Y if X is a parent of Y and X is a female" [8:11]. Facts and rules are collectively referred to as *clauses*.

Consider a small family tree shown in Figure 6 [8:1-26]. Each level of the graph represents a generation. The fact that John is a parent of Ann can be encoded in Prolog as "`parent(john,ann).`". Here `parent` is a name of a relation which connects its two *arguments*, `john` and `ann`. The name of a relation, in this case `parent`, is often referred to as the *predicate*. The family tree shown in Figure 6 can be defined by a program consisting of the following facts:

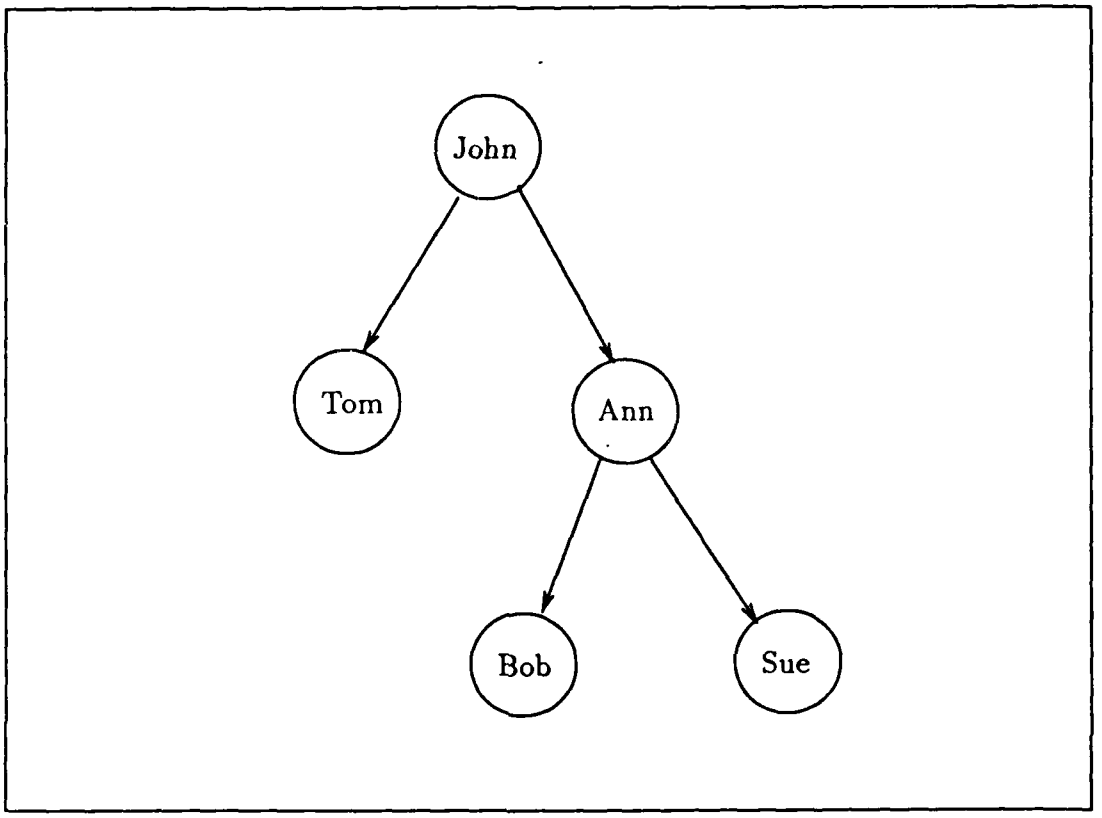


Figure 6. A Family Tree

```
parent(john,tom).
parent(john,ann).
parent(ann,bob).
parent(ann,sue).
```

The Prolog program is put in a database through the *Prolog interpreter.* The interpreter is a program that reads and executes Prolog code. The program is initiated by means of *query.* For example, one such query is, "`Who is Sue's parent?`". In Prolog syntax, the question can be written as "`?- parent(Prnt,sue).`". Prolog returns `ann` as the value of `Prnt`. By convention, the names of predicates and constants begin with a lower-case letter. A variable, such as `Prnt`, representing general

objects begins with an upper-case letter. Each clause terminates with a period.

Now, a question, "`Is Ann a parent of Bob?`", can be asked to the Prolog system by querying "`?- parent(ann,bob).`". The Prolog answer will be `yes` because the fact, "`parent(ann,bob).`", is defined in the database. However, querying, "`?- parent(tom,bob).`", would produce `no` response because nothing is mentioned about Tom being a parent of Bob.

To find solutions, Prolog works by *pattern matching* – matching the variable to a constant value to produce the response. The assignment of a value to a variable is called *instantiation.* Asking the system to find a match is referred to as *satisfying a goal.* Questions to the Prolog system consists of one or more goals. To find out two different children of Ann, the following query can be asked which consists of three goals [7]:

```
?- parent(ann,Firstchild),
   parent(ann,Secondchild),
   Firstchild =/= Secondchild.
```

In searching for a solution, the Prolog tries to satisfy the goals one by one proceeding from the top to bottom. It starts to search for any match at the top of the database that contains the family relation program. When "`parent(ann,bob)`" is encountered in the database, the `Firstchild` is instantiated to `bob` to satisfy the first goal. Now the Prolog tries to satisfy the second goal and instantiates `Secondchild` to `bob` again, because the search always starts from the top of the database unless instructed otherwise. The third goal, which specifies the value of `Firstchild` cannot be the same as the `Secondchild`, fails to be satisfied, since both `Firstchild` and `Secondchild` are instantiated with the same value, `bob`. When this happens the Prolog *backtracks* to the previous goal and uninstantiates the variable and resumes the search for another match. When "`parent(ann,sue).`" is encountered the `Secondchild` is instantiated to `sue`, and satisfies the third goal to give the following solution:

```
Firstchild = bob,
Secondchild = sue
```

A further backtracking to the first goal would produce another solution until no solution can be found.

The program can be extended by using rules and adding more information into the program. To find out "who is a mother of a child", a new relation describing the mother has to be defined. The specification of mother can be stated as a rule, based on the following statement:

```
Mom is the mother of Child if
      Mom is a parent of Child and
      Mom is a female.
```

The above rule can be definded in Prolog as:

```
mother(Mom,Child) :-
    parent(Mom,Child),
    female(Mom).
```

To complete the mother relation, the sex of people has to be defined. This can be accomplished by adding the following facts into the database:

```
male(john).
male(tom).
female(ann).
male(bob).
female(sue).
```

Now asking "?- mother(Mom,sue)." would produce "Mom = ann" after matching Sue's parent as Ann and satisfying Ann is a female.

Built-in features, including the backtracking and pattern matching, make Prolog a very flexible and powerful language. Another useful feature of Prolog, besides

pattern matching and backtracking, is the manipulation of lists [7:85-86, 8:64-78]. Prolog represents a list as a data structure consisting of a first element or *head*, followed by the rest of the elements which constitute the *tail* of the list. A list of elements is represented in the form,`[a,b,c,d,e]`, where `a` is the head , and `[b,c,d,e]` is the tail. Alternatively, a list can be represented in the form `[X|Y]` where `X` represents the head and `Y` represents the tail of the list. Operations involving lists are most conveniently handled using recursive procedures. A detailed discussion on list manipulation and other Prolog programming practices can be found in various Prolog programming books [8, 25].

2.8 Summary

The network performance of a perfectly reliable network (deterministic network) is measured by finding maximum flow through the network. When a network is unreliable (stochastic network), the performance is measured by solving for the expected maximum flow. Unfortunately, determining the exact value of expected maximum flow is intractable. Thus, finding computationally feasible bounds is an attractive alternative way to measure the network performance. Although, the range between upper and lower bound does not represent the true single-valued expected maximum flow, it reflects "a range of flow rerouting choices which are open to the decision maker" [10:452-453]. Furthermore knowing the range is useful, because, in practice, the prediction of exact network performance may be impossible, since " the extent to which rerouting of flow is accomplished may be uncertain" [10:452]. Because of the Prolog language's special features, such as backtracking, search, and list manipulation, Prolog is well suited for solving problems, such as finding paths in the network, that involve an object and relations between objects.

III. Methodology

This chapter discusses the research plan to accomplish the research goal stated in chapter I. This plan consisted of specific tasks performed in a step-by-step fashion. The specific tasks were 1) understanding the problem, 2) formulating the performance models, 3) formulating the investment strategy models, 4) selecting software to solve the formulations, 5) developing Prolog program: FORMULA, and 6) analyzing communication networks.

3.1 Understanding the Problem

The first task was to review the characteristics of deterministic networks and to understand the nature of stochastic networks. While the performance of a deterministic network can be measured in terms of maximum flow through the network, the performance of a stochastic network can be measured in terms of reliability of the network or the expected maximum flow through the network. The reliability ot the network is defined as the probability of having at least one path from source to sink, so that there is some flow between those two nodes. The reliability of the network is a function of network topology and component survival probabilities. The maximum flow of the network is a function of network topology and component capacities. The criteria that affect the expected maximum flow are network topology, component survivabilities, and component capacities. Thus, using the expected maximum flow as a measure of performance incorporates both reliability and maximum flow. Intuitively, some reader might wonder if the expected maximum flow is the product of maximum flow through the network and the reliability of the network. This is, in general, false, because each criterion affects the value of expected maximum flow.

Whether the performance is measured in terms of reliability or expected maximum flow, the calculations required for both measurements are computationally infeasible for a large network. Because they both belong to a so called NP-class which

are, simply stated, "the problems which today can only be solved in exponential time" [24:6]. With current available algorithms and technology, only approximate values can be obtained for a large network in real time. In this research, the bounds of expected maximum flow were investigated. Specifically, the lower bound of expected maximum flow was used as a measure of stochastic network performance, because the empirical results indicated that there was a tendency that the expected maximum flow was much closer to the lower bound than the upper bound. Furthermore, the lower bound was a more significant measure of performance as compared to the upper bound, because it provided the worst expected performance of the network. In designing a new system, knowing this information can be critical. For those who are interested in pursuing network reliability, refer to specific references [4, 5, 18, 20, 27, 32, 34].

3.2 Formulating the Performance Models

The next step was to develop mathematical programming models that measured the performance of the network. Specifically, maximum flow, lower bound of expected maximum flow, and upper bound of expected maximum flow models were needed. The maximum flow provided absolute best performance level of the network, while the bounds provided expected performance range. Using the lower bound as a performance measure, the model not only assessed the network performance, but also identified bottlenecks and significant paths in the system. The theoretical models which find the values of maximum flow and lower and upper bound of expected maximum flow are readily available in numerous textbooks and publications.

When the maximum flow was formulated based on arc-node incidence matrix, as shown in Table 1 in Chapter II, the objective was to maximize the feasible flow from source to sink; the feasible flow was subject to conservation of flow and bounded by arc capacity constraints. The upper bound was formulated in the same manner except it was bounded by expected capacity of the arcs. In the case of lower bound,

the objective was to maximize the sum of expected path flows constrained by arc capacities. Although the maximum flow and upper bound of expected maximum flow could be formulated based on either an arc-node or arc-path incidence matrix, the lower bound could only be formulated using an arc-path incidence matrix. The formulations of these three models are presented in Chapter IV.

3.3 Formulating the Investment Strategy Models

To improve the performance of the network, an investment had to be made in the network. The improvement could be made by increasing the capacities of the components, that is, by expanding the capability of the components to handle the information, or by increasing the survival probabilities of the components. Since, for the purpose of this research, increasing the capacities of components was much easier and cheaper than increasing the survival probabilities of components, the improvements were made in terms of the capacity increases.

Two mathematical programming models were developed to determine the optimal investment strategy that maximized the network performance. The first model was a linear programming model in which the investment decisions were incorporated as continuous variables. These decision variables represented the amount of capacity increases in the components. The second model was a mixed integer programming model in which the investment decisions were incorporated as integer (or zero-one) variables. These integer variables represented the number of investments in the components where the capacity could only be increased in increment of a lump sum amount. The solutions obtained from both models identified exactly in which components and by how much the investment should be made. These models also identified a new performance level when the investment was made. These two models are also presented in Chapter IV.

3.4 Selecting the Right Tool

Whatever the tool used to solve formulated models, the solution, as a minimum, must provide the necessary information discussed in previous sections. The formulated problems could have been solved using any available algorithms, but none of the algorithms could solve all five models and provide all necessary information, such as identifying bottlenecks and significant paths. Hence, off-the-shelf mathematical programming packages were considered which could solve all five models and provide necessary information. Among the packages considered, including SAS, MINOS, and MICROSOLVE, LP/MIP-83 was chosen. The LP/MIP-83 can solve fairly large problems up to one containing approximately 1200 variables, and the solutions can be obtained fairly quickly.

3.5 Development of Prolog Program: FORMULA

Building models that are based on an arc-path incidence matrix take a large amount of effort, because it involves finding all paths from source to sink. Developing lower bound and investment models are even more cumbersome, because they require the calculation of the reliability of each path. To minimize the effort in generating the performance and investment strategy models, the Prolog program, FORMULA, was developed. This program incorporated the depth-first search to find all paths in the network from source to sink. The Prolog language is especially well suited to carry out this search mechanism, because it has a significant advantage over other high-level languages, such as Fortran, due to special built-in features such as backtracking, pattern matching, and list manipulation capability. Using the paths found, the program also generated all performance and investment strategy models. The models generated by this program were in such a format that they could be analyzed by the LP/MIP 83 programming package. Chapter V presents the description of the program.

3.6 Analyzing Communication Networks

Using the models developed, one small example network and three major "realistic" communication networks were analyzed. The example network was used to demonstrate how to build performance and investment strategy models; it also demonstrated ways to analyze the network, including the sensitivity analysis. For each "realistic" network, a series of questions were answered to complete the analysis of the network. In general, the performance of the network was measured, and the optimal investment strategy was determined to improve the network performance. Additionally, the effect of the survivability of a single important node was investigated. The descriptions of the networks are presented in Chapter VI, and the results and analysis are discussed in Chapter VII. The concluding remarks and recommendations for future research are presented in Chapter VIII.

IV. Problem Formulation

This chapter presents linear programming models (formulations) used to assess the performance of the network and mathematical programming models developed to obtain the investment strategies to improve network performance. The models presented in this chapter are Maximum Flow, Lower and Upper Bound of Expected Maximum Flow, and Investment Strategy Model 1 and 2. The examples of various models discussed in this chapter are also presented.

4.1 Maximum Flow Formulation

The maximum flow through the network measures the performance of the network under normal conditions. The components of this particular network has a survival probability of 1, since they are assumed to be 100% reliable. The maximum flow formulation, discussed in Section 2.3, is presented in Table 6. This formulation is based on the arc-node incidence matrix. The objective is to maximize the feasible flows through the network from s to t. The equality constraints enforce the nodal conservation of flow. The capacity constraints for the arcs ensure that the arc flow is nonnegative and the maximum arc flow is limited by a specified amount.

Another way to formulate maximum flow is by using the arc-path incidence matrix. This formulation is presented in Table 7. In this formulation, the objective function, the sum of all path flows from s to t, is maximized. The inequality constraints ensure that the arc capacities do not exceed the specified limit. Of course, the path flow is nonnegative.

Table 6. Maximum Flow Formulation 1

decision variables

V = Feasible flow through the network

x_i = Flow on arc i

parameter

u_i = Capacity of arc i

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$0 \leq x_i \leq u_i \qquad i = 1, 2, ..., n$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

Table 7. Maximum Flow Formulation 2

decision variable

f_j = Flow on path j

parameters

$a_{ij} = 1$ if arc i lies on path j; 0 otherwise

u_i = Capacity of arc i

objective function

Max $\sum_{j=1}^{q} f_j$

subject to

$\sum_{j=1}^{q} a_{ij} f_j \leq u_i \qquad i = 1, 2, ..., n$

$f_j \geq 0 \qquad j = 1, 2, ..., q$

4.2 Lower Bound Formulation

The formulation of the lower bound of expected maximum flow, as applied in this research, measures the performance of the network under adverse conditions. The components of this network are characterized by survival probabilities and capacities. This formulation is based on the arc-path incidence matrix; all paths from source to sink has to be found before the problem can be formulated. The lower bound formulation, discussed in Section 2.5, is presented in Table 8. The objective function is to maximize the sum of the product of path flow and reliability. Inequality constraints ensure that the sum of all path flow in the arc does not exceed the specified limit. There is no upper bound for the path flow. Nonnegativity is assumed for all variables.

Table 8. Lower Bound Formulation

decision variable

f_j = Flow on path j

parameters

$a_{ij} = 1$ if arc i lies on path j; 0 otherwise

R_j = Reliability of path j

u_i = Capacity of arc i

objective function

$$\text{Max} \sum_{j=1}^{q} R_j f_j$$

subject to

$$\sum_{j=1}^{q} a_{ij} f_j \leq u_i \qquad i = 1, 2, ..., n$$

$$f_j \geq 0 \qquad j = 1, 2, ..., q$$

4.3 Upper Bound Formulation

The upper bound formulation provides an optimistic performance level of the network. Measuring both lower and upper bound provides the range of expected performance. The formulation of the upper bound, discussed in Section 2.6, is presented Table 9. This formulation, based on the arc-node incidence matrix, is actually the maximum flow problem with arc capacities replaced by the expected capacities. Another formulation, presented in Table 10, is derived from the arc-path incidence matrix. The formulation is identical to the Maximum Flow Formulation 2, except that the original arc capacities are replaced by the expected capacities.

Table 9. Upper Bound Formulation 1

decision variables

V = Feasible flow through the network

x_i = Flow on arc i

parameters

p_i = Survival probability of arc i

u_i = Capacity of arc i

objective function

Max V

subject to

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = -V$ r=source node

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = 0$ r=intermediate nodes

$\sum_{i\in\alpha} x_i - \sum_{i\in\alpha'} x_i = V$ r=sink node

$0 \leq x_i \leq p_i u_i \qquad i = 1, 2, ..., n$

where α = A set of arcs terminating at node r

α' = A set of arcs initiating at node r

Table 10. Upper Bound Formulation 2

decision variable

f_j = Flow on path j

parameters

$a_{ij} = 1$ if arc i lies on path j; 0 otherwise

$e(u_i)$ = Expected capacity of arc i

objective function

Max $\sum_{j=1}^{q} f_j$

subject to

$\sum_{j=1}^{q} a_{ij} f_j \leq e(u_i)$ $\quad i = 1, 2, ..., n$

$f_j \geq 0$ $\quad j = 1, 2, ..., q$

4.4 Investment Strategy Model 1

The improvement of the network is made by increasing the capability of the components to handle the information. The investment is made to increase the capacity of components such that the lower-bound-performance is optimized. There are two possible approaches to determine the investment strategy: a linear programming model and mixed integer programming model. This section discusses the linear programming model in which the investment decisions are incorporated as continuous variables. The solution obtained from this model identifies which components and how much of the capacity increase should be made. It also provides the optimal performance level when the investment is made. The formulation of this investment model is presented Table 11.

The objective function is the same as that of the lower bound formulation, that is, to maximize the sum of the product of path flow and reliability to yield an improved performance level. The arc inequality constraints ensure that the sum of all path flow in the arc does not exceed the sum of original capacity and any additional capacity due to investment. The total amount of investment in the network is limited by the available investment budget. Again, nonnegativity holds for all variables.

The investment strategy model shown in Table 11 assumes that there is no upper limit on how much capacity can be increased on each component as long as the sum of investment cost in each component does not exceed the total investment budget. If the amount of capacity increase on each component is limited, then the following constraint has to be added in the model:

$$d_i \leq y_i, \qquad i = 1,. \ . \ . \ , n,$$

where y_i is the maximum amount of capacity increase allowed in component i.

Table 11. Investment Strategy Model 1

decision variables

d_i = Amount of capacity increase in arc i

f_j = Flow on path j

parameters

a_{ij} = 1 if arc i lies on path j; 0 otherwise

B = Total budget available for investment

c_i = Cost of increasing the capacity of arc i by one unit

R_j = Reliability of path j

u_i = Capacity of arc i

objective function

$$\text{Max } \sum_{j=1}^{q} R_j f_j$$

subject to

$$\sum_{j=1}^{q} a_{ij} f_j \leq u_i + d_i \qquad i = 1,2,...,n$$

$$\sum_{i=1}^{n} c_i d_i \leq B$$

$$d_i \geq 0 \qquad i = 1,2,...,n$$

$$f_j \geq 0 \qquad j = 1,2,...,q$$

4.5 Investment Strategy Model 2

This section discusses the second model, a mixed integer programming model, developed to determine the optimal investment strategy. In this model, the investment decisions are incorporated as integer variables, corresponding to the number of investments that can be made In each investment, only a predetermined lump sum amount of capacities can be increased, so that each investment makes a step-wise improvement in the component capacities. The solution obtained from this model identifies where the investment should be made and by how much; it also provides the improved performance level when the investment is made. The formulation of this investment strategy model is shown in Table 12. The objective here is, again, to maximize the sum of expected path flows. The explanation of constraint function is analogous to that of Investment Strategy Model 1. The sum of all path flow in the arc is constrained by the sum of original capacity and additional capacity due to investment. The total amount of investment is constrained again by the total investment budget. The path flow variables, f_j, are nonnegative, but the investment decision variables, g_i, can only assume integer value.

The investment decision variables can also be incorporated as *zero-one* variables. In this case, the *zero* corresponds to the decision not to invest and *one* corresponds to the decision to invest in the component. Thus, zero-one decision variables allow only one investment to be made by a predetermined amount.

Table 12. Investment Strategy Model 2

decision variables

f_j = Flow on path j

g_i = Number of investments in component i

parameters

a_{ij} = 1 if arc i lies on path j; 0 otherwise

B = Total budget available for investment

b_i = Predetermined amount of capacity increase, per investment, in arc i

c_i = Cost of increasing the capacity of arc i by one unit

R_j = Reliability of path j

u_i = Capacity of arc i

objective function

$$\text{Max } \sum_{j=1}^{q} R_j f_j$$

subject to

$$\sum_{j=1}^{q} a_{ij} f_j \leq u_i + b_i g_i \qquad i = 1, 2, ..., n$$

$$\sum_{i=1}^{n} b_i c_i g_i \leq B$$

$$g_i = \text{integer} \qquad i = 1, 2, ..., n$$

$$f_j \geq 0 \qquad j = 1, 2, ..., q$$

4.6 Examples

The purpose of this section is to show how a network problem can be formulated using the models discussed in this chapter. An example of a stochastic network is shown in Figure 7. It has three stochastic components: arcs 2, 5, and 6. To find the maximum flow through the network, these components are assumed perfectly reliable, so that the probability of survival is 1 for all the components. Using the arc-node incidence matrix, the maximum flow can be formulated as shown in Table 13.

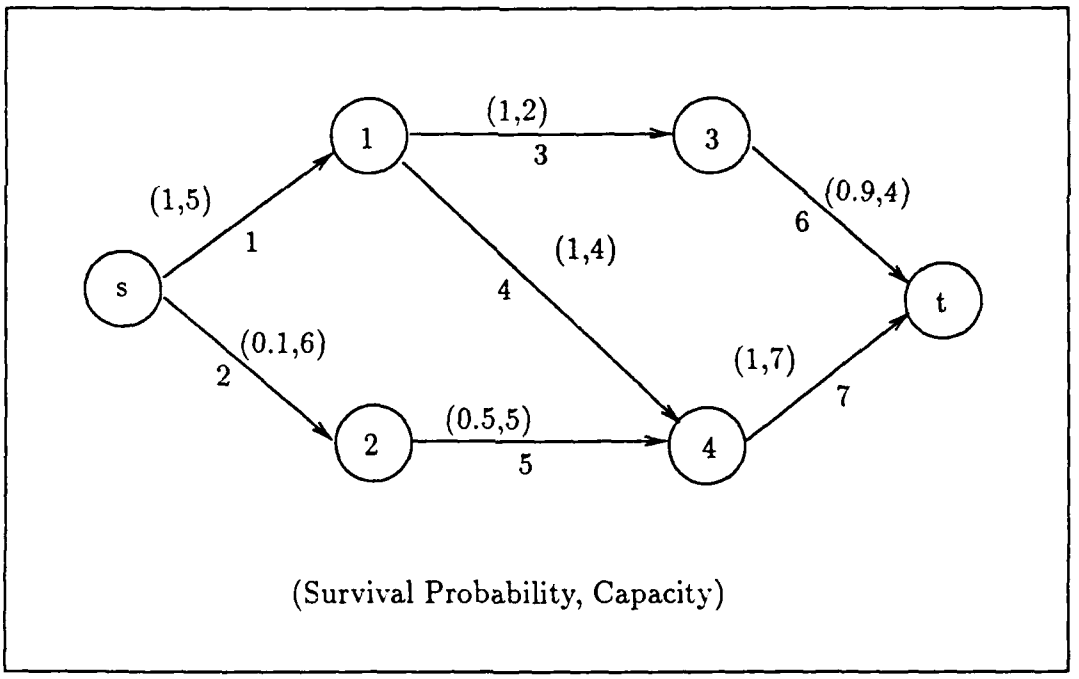


Figure 7. Example Network

Table 13. An Example of Maximum Flow Formulation

objective function

Max V

subject to

$V - x_1 - x_2 = 0$	node s
$x_1 - x_3 - x_4 = 0$	node 1
$x_2 - x_5 = 0$	node 2
$x_3 - x_6 = 0$	node 3
$x_4 + x_5 - x_7 = 0$	node 4
$x_6 + x_7 - V = 0$	node t

$0 \leq x_1 \leq 5, \quad 0 \leq x_2 \leq 6, \quad 0 \leq x_3 \leq 2$

$0 \leq x_4 \leq 4, \quad 0 \leq x_5 \leq 5, \quad 0 \leq x_6 \leq 4$

$0 \leq x_7 \leq 7$

The network shown in Figure 7 has three paths from s to t. Path 1 consists of arcs 1, 3, and 6; path 2 consists of arcs 1, 4, and 7; and path 3 consists of arcs 2, 5, and 7. The reliability of path 1 is $(1)(1)(.9) = 0.9$, path 2 is 1, and path 3 is 0.05. Using the general formulation described earlier, the lower bound can be formulated as shown in Table 14. In Table 14, the path flow is constrained by the minimum arc capacity in the path. Thus, constraints for arcs 2 and 6 (with an astrick) may be deleted since they are redundant. If the reliabilities of the paths are assumed to be 1, that is, the objective function is $f_1 + f_2 + f_3$, then this formulation is the maximum flow formulation that is based on the arc-path incidence matrix.

The upper bound of the expected maximum flow for the network shown in

Table 14. An Example of Lower Bound of Expected Maximum Flow

objective function

$$0.9f_1 + 1f_2 + 0.05f_3$$

subject to

$f_1 + f_2 \leq 5$	arc 1
$f_3 \leq 6$	arc 2*
$f_1 \leq 2$	arc 3
$f_2 \leq 4$	arc 4
$f_3 \leq 5$	arc 5
$f_1 \leq 4$	arc 6*
$f_2 + f_3 \leq 7$	arc 7

$$f_1, f_2, f_3 \geq 0$$

Figure 7 can be found by replacing the original arc capacities with expected arc capacities. The formulation, based on the arc-node incidence matrix, is shown in Table 15.

The performance of the network can be improved by investing in components to increase the capacities to handle the flow. Assumming the cost, c_i, of increasing the capacity by one unit and budget available, B, are as follows: $c_1 = 50$, $c_2 = 60$, $c_3 = 20$, $c_4 = 40$, $c_5 = 50$, $c_6 = 40$, $c_7 = 70$, and $B = 1000$, the investment strategy model can be formulated as shown in Table 16. Note that the decision variables are moved over to the left of the inequality signs to have the formulation in standard format.

Table 15. An Example of Upper Bound of Expected Maximum Flow

objective function

Max V

subject to

$V - x_1 - x_2 = 0$	node s
$x_1 - x_3 - x_4 = 0$	node 1
$x_2 - x_5 = 0$	node 2
$x_3 - x_6 = 0$	node 3
$x_4 + x_5 - x_7 = 0$	node 4
$x_6 + x_7 - V = 0$	node t

$0 \leq x_1 \leq 5, \quad 0 \leq x_2 \leq 0.6, \quad 0 \leq x_3 \leq 2$

$0 \leq x_4 \leq 4, \quad 0 \leq x_5 \leq 2.5, \quad 0 \leq x_6 \leq 3.6$

$0 \leq x_7 \leq 7$

For the second investment model, the capacity of each component can only be increased in increment of predetermined amount. Assume predetermined amount of capacity increase and total budget are proposed as follows: $b_1 = 5$, $b_2 = 5$, $b_3 = 5$, $b_4 = 5$, $b_5 = 5$, $b_6 = 5$, $b_7 = 5$, and $B = 1000$. Then the total cost of one time investment in the component i is $c_i b_i$, where the value of c_i is assumed the same as before. So, as an example, the cost of improving the component 1 (arc 1) is (50)(5) = 250 if investing in arc 1 once. The problem can be formulated as shown in Table 17. The solutions to the models developed in this section will be discussed in Chapter VII, Results and Analysis.

Table 16. An Example of Investment Strategy Model 1

objective function

$$0.9f_1 + 1f_2 + 0.05f_3$$

subject to

$f_1 + f_2 - d_1 \leq 5$	arc 1
$f_3 - d_2 \leq 6$	arc 2
$f_1 - d_3 \leq 2$	arc 3
$f_2 - d_4 \leq 4$	arc 4
$f_3 - d_5 \leq 5$	arc 5
$f_1 - d_6 \leq 4$	arc 6
$f_2 + f_3 - d_7 \leq 7$	arc 7
$50d_1 + 60d_2 + 20d_3 + 40d_4 +$	
$50d_5 + 40d_6 + 70d_7 \leq 1000$	budget

$$d_1, d_2, d_3, d_4, d_5, d_6, d_7 \geq 0$$

$$f_1, f_2, f_3 \geq 0$$

Table 17. An Example of Investment Strategy Model 2

objective function

$$0.9f_1 + 1f_2 + 0.05f_3$$

subject to

$f_1 + f_2 - 5g_1 \leq 5$	arc 1
$f_3 - 5g_2 \leq 6$	arc 2
$f_1 - 5g_3 \leq 2$	arc 3
$f_2 - 5g_4 \leq 4$	arc 4
$f_3 - 5g_5 \leq 5$	arc 5
$f_1 - 5g_6 \leq 4$	arc 6
$f_2 + f_3 - 5g_7 \leq 7$	arc 7

$$250g_1 + 300g_2 + 100g_3 + 200g_4 +$$
$$250g_5 + 200g_6 + 350g_7 \leq 1000 \qquad \text{budget}$$

$$g_1, g_2, g_3, g_4, g_5, g_6, g_7 = integer$$

$$f_1, f_2, f_3 \geq 0$$

V. *The Prolog Program: FORMULA*

This chapter describes the computer program, FORMULA version 1.0, which was written in Prolog to carry out the major portion of this research. General formulation of the model that is based on the arc-path incidence matrix is fairly straight forward in concept; the formulation can easily be obtained for a small network. However, for a large network, generating this formulation can be very cumbersome, because it requires finding all paths in the network from source to sink and determining which paths go through each arc. For lower bound and investment models, it would be even more cumbersome, since it requires calculating the reliabilities of all paths as well. The FORMULA was written to minimize the effort in formulating the models that are based on the arc-path incidence matrix. This program does the following six functions:

- Finds all paths from source(s) to sink(t), and calculates path reliabilities.
- Generates the formulation of maximum flow.
- Generates the formulation of lower bound of expected maximum flow.
- Generates the formulation of upper bound of expected maximum flow.
- Generates the formulation of investment strategy model 1.
- Generates the formulation of investment strategy model 2.

The formulations (or models) obtained from this program serve as a direct input file to off-the-shelf mathematical programming package, LP/MIP-83 linear and mixed integer programming system. A copy of the program is attached in Appendix A. The FORMULA User's Manual, in Appendix B, describes how to prepare an input data file and use the program.

5.1 Input

The input data describes the network to be analyzed. It consists of six data sets: the description of an arc relationship with respect to one another, the survival

probability of each arc, the capacity of each arc, the cost of improving each arc by one unit of capacity, the predetermined amount of capacity increase in each component, and the budget available to improve the network. In Prolog, these inputs are described as facts as follows:

Input	Description
`arc(Arc1,Arc2).`	`Arc1 is the parent of Arc2 (Arc1 precedes Arc2).`
`prob(A,Pb).`	`The survival probability of arc A is Pb.`
`cap(A,Cp).`	`The capacity of arc A is Cp. The unlimited (infinite) capacity is denoted by '*'.`
`cost(A,Cs).`	`The cost of increasing one unit of capacity in arc A is Cs.`
`invest(A,Am).`	`The predetermined amount of capacity increase for arc A is Am.`
`budget(B).`	`Total budget available for investment is B.`

The network described in the input data file must be represented as a network containing a single source s and single sink t. Thus, if a network contains multiple sources and sinks, all sources must be connected to the artificial source s, and all sinks must be connected to the artificial sink t. Additionally, if a network contains stochastic and/or capacitated nodes, the nodes have to be converted to arcs as explained in Chapter II.

5.2 Description of the Program

The program, FORMULA, was written in Prolog using Arity/Prolog version 5.0 [2, 3]. Arity/Prolog is one of the most powerful Prolog packages available on

the market; it has more stack memory space and useful features compared to Prolog 86 or C-Prolog. The program consists of three main parts, each part generating a particular output as requested by the user: paths and reliabilities, performance models, and investment strategy models.

5.2.1 Part One: Finding Paths and Reliabilities To find all paths from source (s) to sink (t), a depth-first search technique was implemented. A depth-first search is well suited to finding paths in the network. The basic idea of this search technique is best explained by using an example. A directed graph is shown in Figure 8. In a depth-first search, the search starts at s and goes down to the next level picking the leftmost child of s. If the node chosen is not the desired node, the search process moves downward, again, picking the leftmost child of the node. "If it reaches the bottom level without finding the desired choice, the process returns to the last node where there was a choice. Then the downward motion is repeated" [21:165]. To reach the goal t, the depth-first search goes through nodes, *1*, *3*, and *7*, down from starting node s. It returns from *7* to *3* and moves down to *8*. Since *8* is not the desired choice, the search process returns to *1*. From node 1, the search goes down to *4* and get to t.

In the program, FORMULA, the depth-first search is implemented as follows:

```
depth_first(Path,Goal,Path) :-
    satisfies(Path,Goal).

depth_first([X|Rest],Goal,Path) :-
    arc(X,Y),
    not member(Y,[X|Rest]),
    depth_first([Y,X|Rest],Goal,Path).
```

The predicates, `satisfies(Path,Goal)` and `member(Y,[X|Rest])` are predefined elsewhere in the program. The `arc(X,Y)` predicate is read in from the input

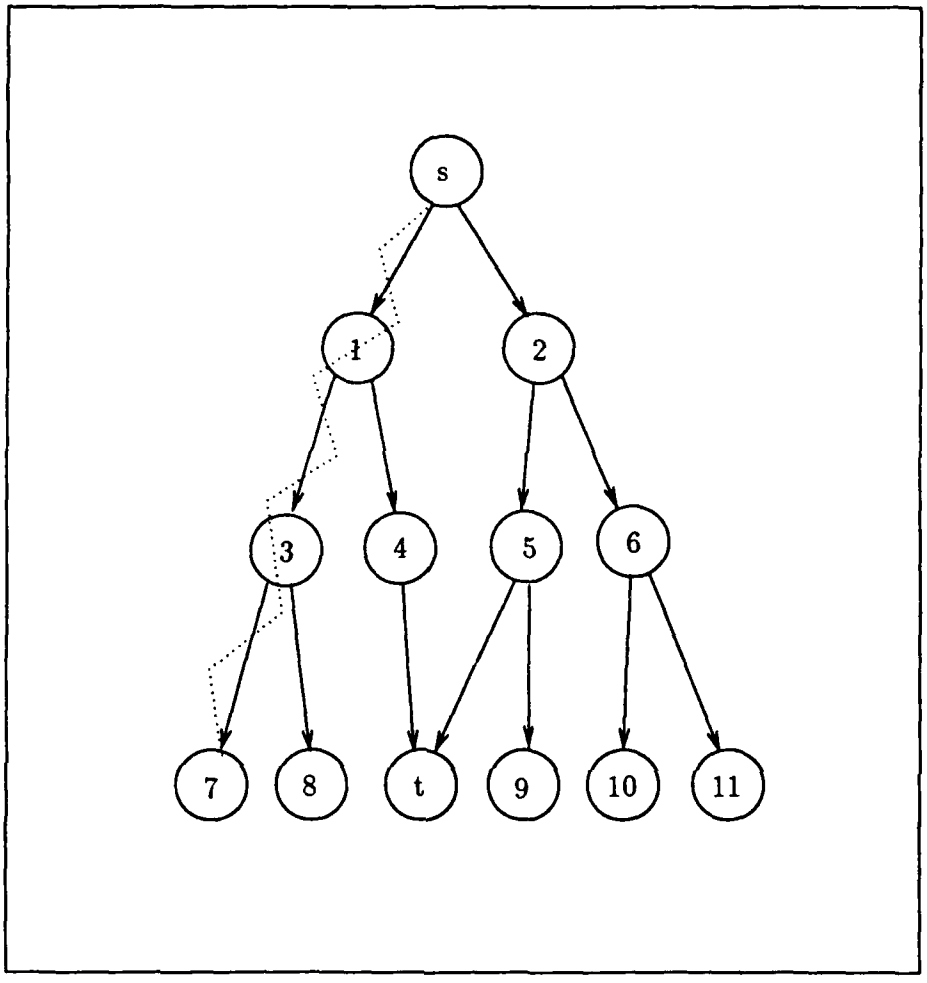


Figure 8. Depth-First Search

data. The depth-first search requires "a recursive procedure in which the recursion occurs for movement down one [arc] and then across all the [arcs]" until a path from s to t is found [21:166]. A path consists of a starting node s, a sink node t, and a list of arcs that connects s and t. To find the remaining paths, another predicate is needed to execute the `depth_first` repeatedly.

The reliability of a path is obtained by simply multiplying all survival probabilities of the arcs that form a path. The reliability is calculated by using a recursive procedure. In each call to itself, a new survival probability of the arc in the path is

read in from the input data and multiplied to previously obtained partial reliability until all arc survivability has been multiplied to get the reliability of the path. The program code which accomplishes this is shown below:

```
find_reliability([],1).
find_reliability([Arc|Rest],Rel) :-
    find_reliability(Rest,RelRest),
    prob(Arc,Pb),
    Rel is Pb * RelRest.
```

5.2.2 Part Two: Performance Models The second part of the program generates the linear programming formulations of the maximum flow and lower and upper bound of expected maximum flow. The formulations consist of a title, objective function, and constraints. The formulation begins with a title statement that describes the model. The objective function is generated by finding all paths and calculating the path reliabilities using the features of part one described in the previous section. Each term of objective function is a product of the parameter (reliability) and the decision variable (path flow). For maximum flow and upper bound formulations, the path reliabilities are 1. The main Prolog code that generates these formulations are shown below:

```
objective(Selection) :-
      .
      .
   find_objective(s,t,Selection).


find_objective(Start,Goal,Selection) :-
   depth_first([Start],Goal,Path),
   find_reliability(Path,Rel),
   ctr_inc(2,Pnbr),
   output_objective(Pnbr,Rel,Selection),
   make_arc_path_matrix(Path,Pnbr),
   fail.

find_objective(_,_,_).
```

The argument `Selection` in the predicate `find_objective(Start,Goal, Selection)` determines whether the formulation to be generated is maximum flow, lower bound, or upper bound. The value of `Selection` is supplied by the user. The constraint functions, which are arc capacity constraints, are generated from the arc-path incidence matrix. The path is converted to an arc-path incidence matrix by the `make_arc_path_matrix (Path,Pnbr)` predicate. The matrix is represeted as `arc_path_matrix(Arc,Path_Number)` in the database. For example, the following small part of matrix implemented in the database represents arcs 4, 10, 45, 55, and 65 form the path 164:

```
.
.
arc_path_matrix(4,164).
arc_path_matrix(10,164).
arc_path_matrix(45,164).
arc_path_matrix(55,164).
arc_path_matrix(65,164).
.
.
```

For each arc with a finite capacity, the program goes through the whole matrix and finds all paths that contain the particular arc of interest and output them through the `output_constraints` predicate:

```
constraint :-
     find_all_arc_lists(Arc_List),
          .
          .
     output_constraints.
```

5.2.3 Part Three: Investment Strategy Models The third part generates the investment strategy models that improve the performance by increasing the lower bound. The process of generating these formulations is similar to that of the performance models described in the previous section. The only difference is that the

investment decision variables are added in the constraint inequalites and a new investment budget constraint is introduced to limit the investments.

5.3 Output

The program FORMULA generates six sets of output: paths and their reliabilities, maximum flow formulation, lower bound formulation, upper bound formulation, investment strategy model 1, and investment strategy model 2; only one output is generated at a time as requested by the user. The output can be either displayed on the monitor screen or sent to an output file. When the user requests the output to be sent to a file, the output of paths and reliabilities is sent to 'output1.lp', maximum flow to 'output2.lp', lower bound to 'output3.lp', upper bound to 'output4.lp', investment strategy model 1 to 'output5.lp', and investment strategy model 2 to 'output6.lp'. The output files, 'output2.lp' through 'output6.lp', serve as a direct input file to LP/MIP-83 for further analysis.

VI. Description of Communication Networks

This chapter contains the description of the example network shown in Figure 7 (in Chapter IV) and the three "realistic" communication networks analyzed in this research. In the next chapter, the example network will be used to demonstrate a complete analysis methodology, including the sensitivity analysis; the three "realistic" networks will be analyzed as case studies. These three networks are arbitrarily named Network A, B, and C. For each network, a brief description of the network, the topology of the network, and the characteristics of the components are presented.

6.1 Example Network

The example network, shown in Figure 7, contains 6 nodes and 7 arcs; it has a single source and single sink. All components are perfectly reliable, except three arcs, arcs 2, 5, and 6, which are stochastic. This network contains three paths from source to sink. Path 2, which consists of arcs 1, 4, and 7, has a reliability of 1. Because of this perfect reliability, path 2 will most likely to carry the majority of flows, and arc 4, which has the lowest capacity in path 2, will likely to become the bottleneck.

6.2 Network A

The topology of the Network A is shown in Figure 9, and the characteristics of the arcs and nodes, in terms of survival probability and capacity, are described in Tables 18 and 19. This network contains 19 nodes and 27 arcs; it has multiple source nodes and a single sink node. The network contains dependent components: four pairs of nodes and six pairs of arcs. These pairs are shown in Tables 20 and 21. Two dependent components that form a pair actually represent a single physical medium. This medium is represented twice to avoid cycles in the network. Thus, two dependent components that form a pair are perfectly positively correlated; that

is, if one fails, the other one also fails. For example, in Table 20, if the node 15 fails, then node 16 fails simultaneously. Hence, the probability of the components in a dependent pair must have the same probability.

Looking at the topology, all paths from source to sink go through node 14. If this node fails, there will be no flow at all from source to sink. The survival probability of this node will affect the performance of the network significantly. All arcs leaving the source can only handle the flow up to 1200 units. Since the majority of the remaining arcs can handle up to 4800 units, the arcs leaving the source will most likely to form the bottlenecks in the system.

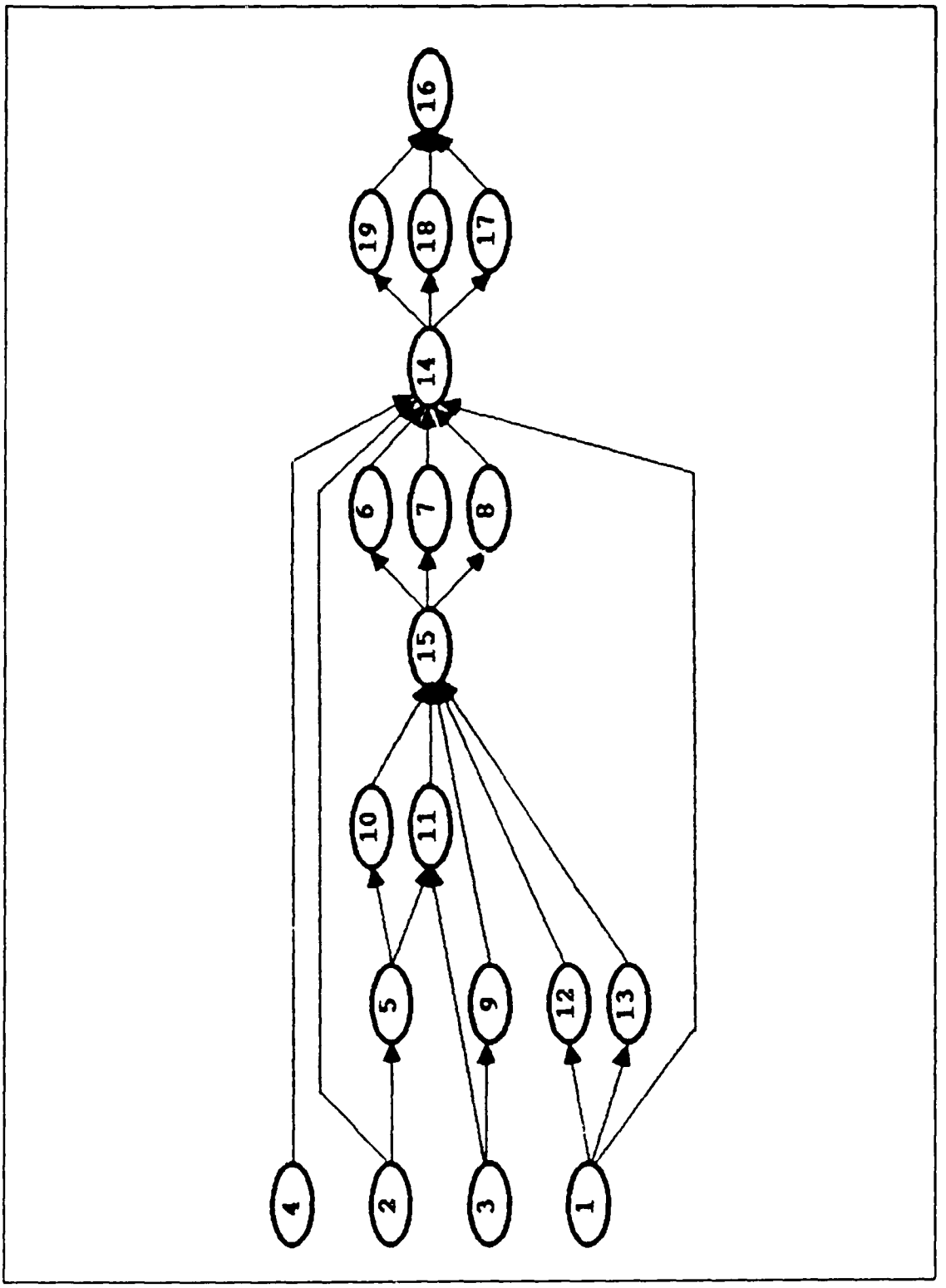


Figure 9. Topology of Network A

Table 18. Description of Arcs in Network A

Arcs			
Initial Node	Terminal Node	Survival Probability	Capacity
1	12	1.0	1200
1	13	1.0	1200
1	14	1.0	1200
2	5	0.3	1200
2	14	0.6	1200
3	9	1.0	1200
3	11	1.0	1200
4	14	1.0	1200
5	10	0.6	1200
5	11	0.7	1200
6	14	0.6	4800
7	14	0.6	4800
8	14	0.3	4800
9	15	1.0	4800
10	15	0.6	4800
11	15	1.0	4800
12	15	0.7	4800
13	15	1.0	4800
14	17	0.3	4800
14	18	0.6	4800
14	19	0.6	4800
15	6	0.3	4800
15	7	0.6	4800
15	8	0.7	4800
17	16	0.7	4800
18	16	0.6	4800
19	16	0.3	4800

Table 19. Description of Nodes in Network A

Node	Survival Probability	Capacity
1	1.0	*
2	0.3	*
3	0.7	*
4	0.5	*
5	0.8	*
6	1.0	*
7	0.3	*
8	0.7	*
9	0.5	*
10	0.8	*
11	1.0	*
12	0.3	*
13	0.7	*
14	0.5	*
15	0.8	*
16	0.8	*
17	0.7	*
18	0.3	*
19	1.0	*

* All nodes have unbounded (infinite) capacity

Table 20. Dependent Nodes in Network A

node 8	and	node 17
node 6	and	node 19
node 7	and	node 18
node 15	and	node 16

Table 21. Dependent Arcs in Network A

arc 6,14	and	arc 14,19
arc 7,14	and	arc 14,18
arc 8,14	and	arc 14,17
arc 15,6	and	arc 19,16
arc 15,7	and	arc 18,16
arc 15,8	and	arc 17,16

6.3 Network B

The topology of the Network B is shown in Figure 10, and the characteristics are described in Tables 22 and 23. This network consists of 26 nodes and 36 arcs, and contains multiple sources and sinks. All component failures are independent. Since alll nodes are non-capacitated, they can handle any amount of flow that comes through. All arcs are stochastic, but only half of the nodes are stochastic.

In this topology, node 16 seems most important, since a majority of the paths from source to sink go through this node. It has a survival probability of 0.07, which indicates that this node is down 93% of the time. The arcs starting from node 16 have a capacity of only 75. These arcs will probably form the bottlenecks. There are three direct paths, which consist of only one arc, from source to sink; they have at least a 0.5 survival probability. Because of these high survival probabilities and their low capacities, these paths will most likely to form the bottlenecks as well. In general, the components with high survival probabilities and low capacities tend to become the bottlenecks in the system.

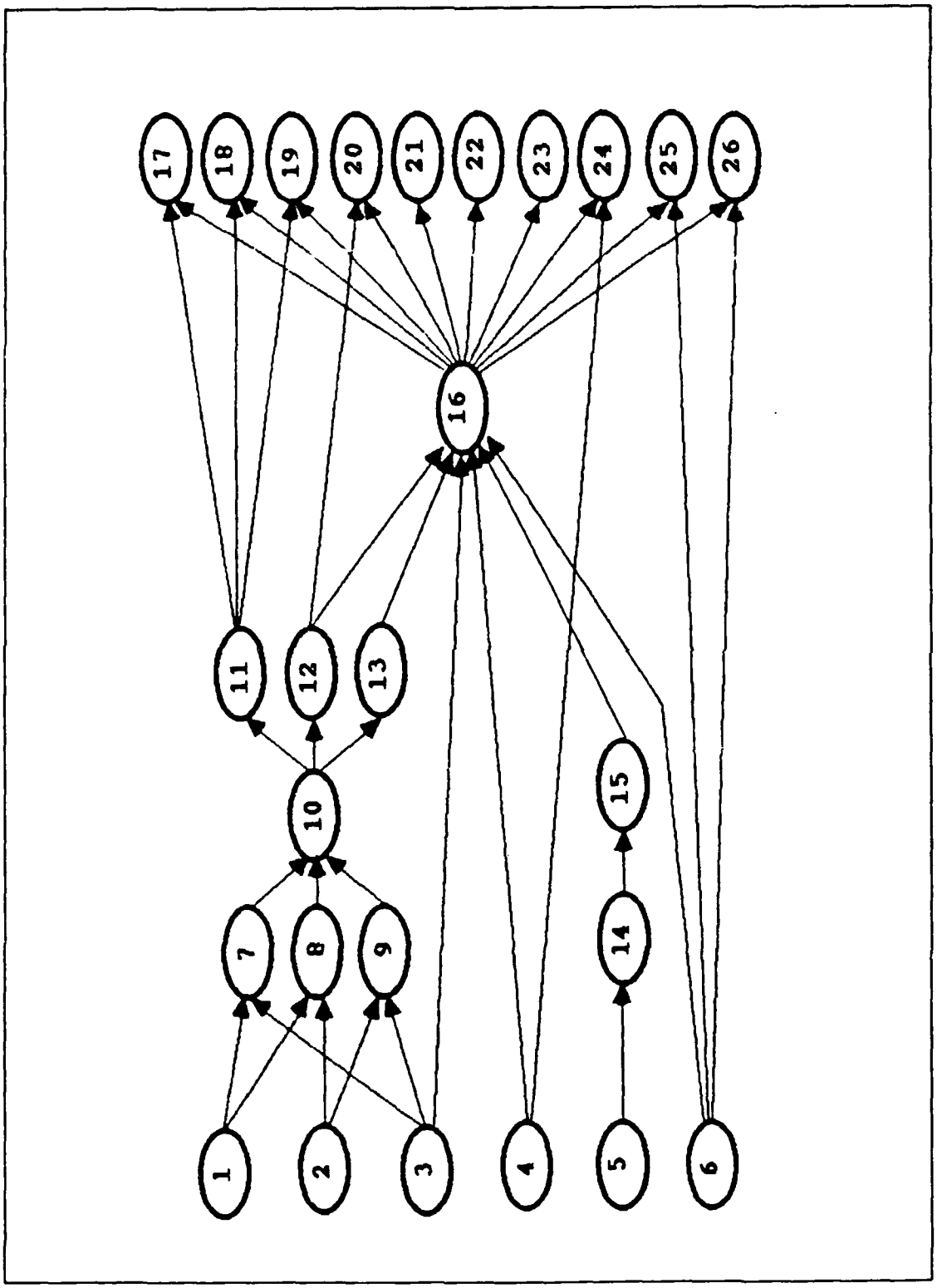


Figure 10. Topology of Network B

Table 22. Description of Arcs in Network B

Arcs			
Initial Node	Terminal Node	Survival Probability	Capacity
1	7	0.8	150
1	8	0.8	200
2	8	0.5	750
2	9	0.5	750
3	7	0.8	200
3	9	0.5	750
3	16	0.6	150
4	16	0.8	200
4	24	0.5	600
5	14	0.8	1200
6	16	0.5	1200
6	25	0.6	75
6	26	0.6	75
7	10	0.5	1200
8	10	0.7	1200
9	10	0.5	2400
10	11	0.5	1200
10	12	0.7	1200
10	13	0.7	1200
11	17	0.5	1200
11	18	0.5	75
11	19	0.5	1200
12	16	0.7	1200
12	20	0.5	1200
13	16	0.7	600
14	15	0.8	1200
15	16	0.8	1200
16	17	0.6	75
16	18	0.6	75
16	19	0.6	75
16	20	0.6	75
16	21	0.6	75
16	22	0.6	75
16	23	0.6	75
16	24	0.6	75
16	25	0.6	75
16	26	0.6	75

Table 23. Description of Nodes in Network B

Node	Survival Probability	Capacity
1	0.70	*
2	0.15	*
3	0.03	*
4	1.00	*
5	1.00	*
6	0.04	*
7	0.40	*
8	1.00	*
9	0.01	*
10	0.70	*
11	0.11	*
12	1.00	*
13	0.06	*
14	0.09	*
15	0.18	*
16	0.07	*
17	1.00	*
18	1.00	*
19	1.00	*
20	1.00	*
21	1.00	*
22	1.00	*
23	1.00	*
24	1.00	*
25	1.00	*
26	1.00	*

* All nodes have unbounded (infinite) capacity

6.4 Network C

The topology of the Network C is shown in Figure 11, and the component characteristics are shown in Tables 24 and 25. This network consists of 39 nodes and 54 arcs. This network contains multiple sources and sinks. All component failures are independent of one another.

Looking at the topology, all paths go through node 11. The survival probability of this node would greatly affect the performance of the network, because if this node fails, all paths will fail. Distribution of survival probabilities and capacities indicate the majority of arcs prior to node 11 would probably cause bottlenecks in the system.

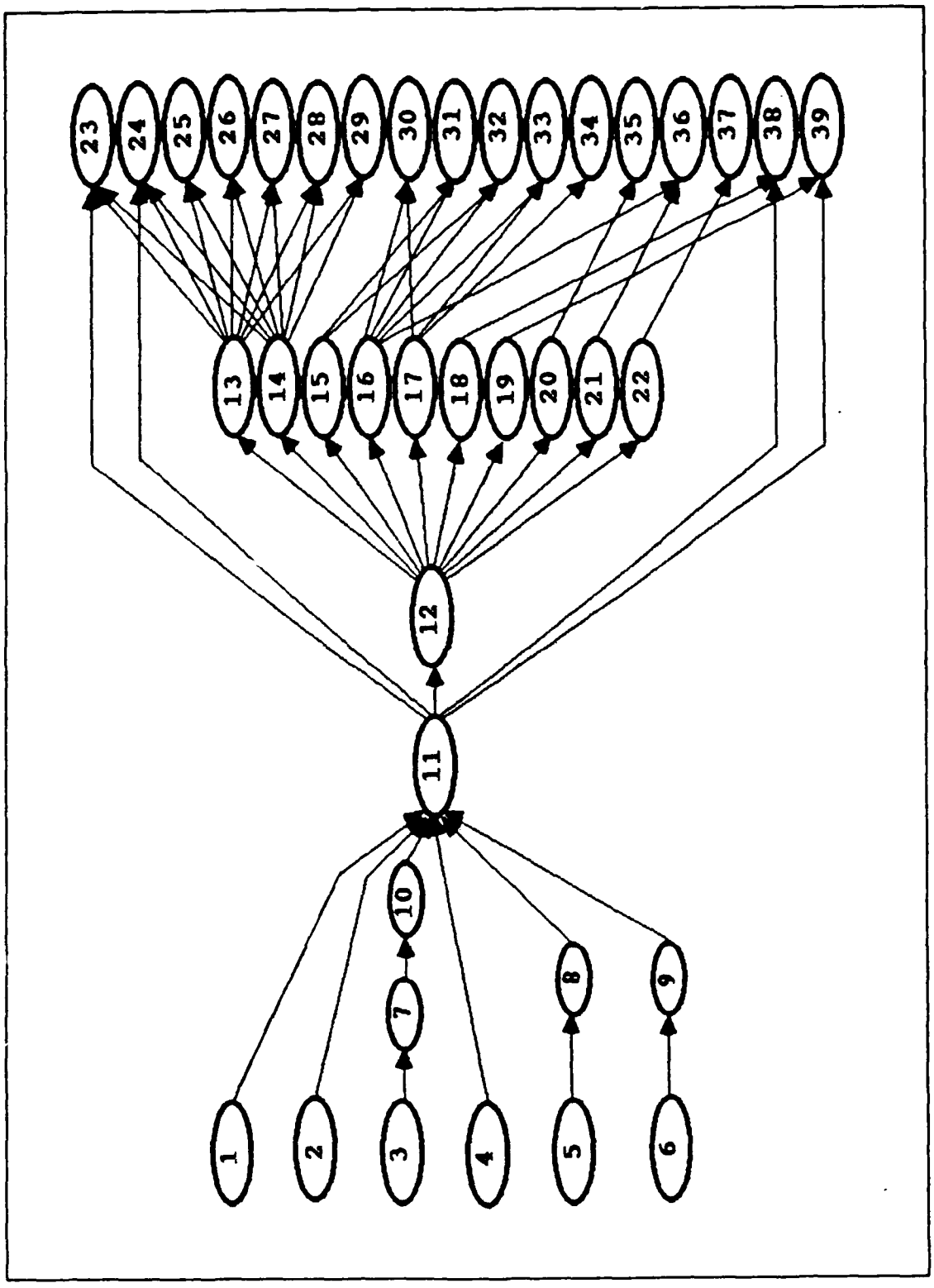


Figure 11. Topology of Network C

Table 24. Description of Arcs in Network C

Arc				Arc			
Init Node	Term Node	Survival Prob	Capacity	Init Node	Term Node	Survival Prob	Capacity
1	11	0.9	1200	13	25	0.7	2400
2	11	1.0	1200	13	26	0.9	1200
3	7	1.0	300	13	27	1.0	1200
4	11	0.6	1200	13	28	1.0	1200
5	8	0.3	1200	13	29	0.3	1200
6	9	0.6	1200	14	23	0.6	4800
7	10	1.0	300	14	24	0.3	4800
8	11	0.9	1200	14	25	0.6	2400
9	11	1.0	1200	14	26	0.7	1200
10	11	0.7	300	14	27	0.9	1200
11	12	0.9	9600	14	28	1.0	1200
11	23	0.6	75	14	29	1.0	1200
11	24	0.3	75	15	31	0.6	2400
11	38	0.6	1200	15	32	0.3	1200
11	39	0.7	1200	16	30	0.6	300
12	13	1.0	4800	16	31	0.7	2400
12	14	1.0	4800	16	32	0.9	1200
12	15	0.6	4800	16	33	1.0	300
12	16	0.3	4800	16	36	1.0	2400
12	17	0.6	4800	17	30	0.6	1200
12	18	0.7	2400	17	33	0.3	300
12	19	0.9	4800	17	34	0.6	300
12	20	1.0	4800	18	38	0.7	1200
12	21	1.0	4800	19	39	0.9	1200
12	22	0.6	2400	20	35	1.0	2400
13	23	0.3	4800	21	36	1.0	2400
13	24	0.6	4800	22	37	0.6	600

Table 25. Description of Nodes in Network C

Node	Survival Probability	Capacity	Node	Survival Probability	Capacity
1	0.3	*	21	0.3	*
2	0.7	*	22	0.7	*
3	0.5	*	23	0.5	*
4	0.8	*	24	0.8	*
5	1.0	*	25	1.0	*
6	0.3	*	26	0.3	*
7	0.7	*	27	0.7	*
8	0.5	*	28	0.5	*
9	0.8	*	29	0.8	*
10	1.0	*	30	1.0	*
11	0.7	*	31	0.3	*
12	0.7	*	32	0.7	*
13	0.5	*	33	0.5	*
14	0.8	*	34	0.8	*
15	1.0	*	35	1.0	*
16	0.3	*	36	0.3	*
17	0.7	*	37	0.7	*
18	0.5	*	38	0.5	*
19	0.8	*	39	0.8	*
20	1.0	*			

* The node has unbounded (infinite) capacity.

VII. Results and Analysis

This chapter discusses the analysis of the example network and the three communication networks described in Chapter VI. The example network was used to demonstrate a complete analysis methodology, including the sensitivity analysis; the three "realistic" networks were analyzed as a case study. For each "realistic" network, a series of questions were answered to complete the analysis. Except for the example network, the original networks were revised so that they contained a single source and single sink named s and t, respectively. Additionally, all capacitated or stochastic nodes were represented as dummy arcs with two nodes as explained in Chapter II. All performance and investment strategy models were developed based on these revised networks. This way, the comparison of formulations and solutions among the models could be made easy. The performance and investment strategy models were generated by the FORMULA computer program. The models were solved using the LP/MIP-83. The experimental results for the example network and Network A are attached in Appendices C and D, respectively. The results for the other two networks are omitted, but the revised topologies for these networks are included in Appendices E and F, from which the models were developed.

7.1 Analysis of Example Network

For the example network shown in Figure 7, there was no need to revise the network, since it was already in analyzable form. The input data, which describes this network, for the FORMULA program is shown in Appendix C.1. Appendix C.2 contains a listing of all paths and their reliabilities from s to t. Appendices C.3, C.4, and C.5 contain solutions to maximum flow, lower bound, and upper bound models, respectively. The performance of the example network is summarized in Figure 12. The maximum flow through this network was 9 which is the absolute

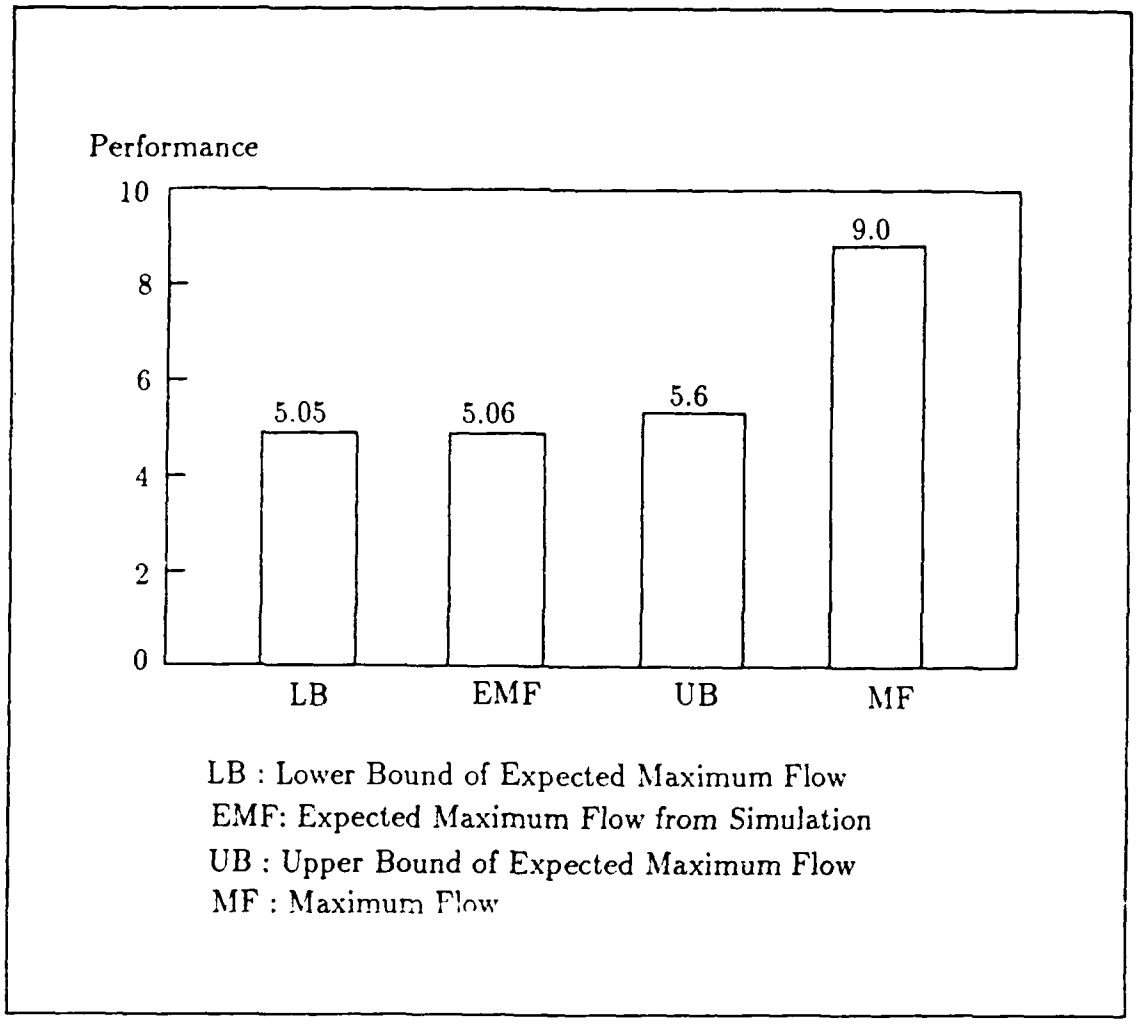


Figure 12. Performance Level of Example Network

best performance level of this network. The exact value of expected maximum flow[1] was 5.06. The range of the expected maximum flow was from 5.05 to 5.6.

According to the solution to the lower bound of expected maximum flow shown in Appendix C.4, all three paths carried the flow as described below:

[1]The expected maximum flow was calculated by enumerating all 8 possible failure states as discussed in Chapter II.

Number	Path	Path Reliability	Path Flow	Expected Path Flow
1	path 1*: s,1,3,t	0.9	1.0	0.9
2	path 2 : s,1,4,t	1.0	4.0	4.0
3	path 3 : s,2,4,t	0.05	3.0	0.15

* Path 1 consists of nodes s, 1, 3, and t.

The expected path flow was obtained by multiplying the path reliability and path flow. Path 2 had the highest expected path flow, which made it most significant. The sum of the expected path flows was the lower-bound-performance level.

The bottlenecks were arcs 1, 4, and 7. They are identified, with thick lines, in Figure 13.

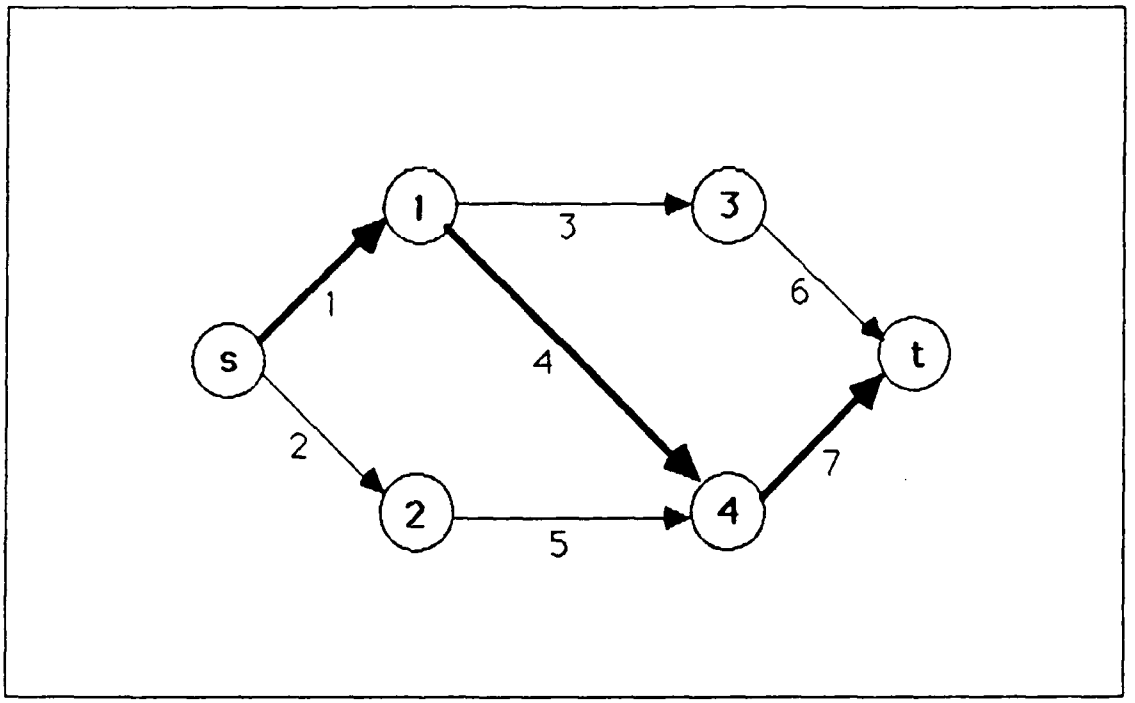


Figure 13. Bottleneck Arcs in Example Network

The sensitivity analysis was performed to investigate the effects of single changes of the parameters without re-solving the problem. The sensitivity analysis centered

on two distinct areas: the coefficients of the variables in the objective function and the values of the right hand side of the constraint inequalities. The coefficients of the variables corresponded to path reliabilities, and the values of the right hand side corresponded to component capacities. The results of sensitivity analysis, performed using the LP/MIP-83, for the lower bound of expected maximum flow are shown in Appendix C.4.

From Appendix C.4, the reliability of path 1 could have been any value between 0.0 and 0.95 and still maintain the same network performance level, as long as changing the reliability of this path does not affect the reliabilities of the other paths. When this reliability reaches 0.0, the flow on arc 1 would be effected. On the other hand, when it reaches 0.95, the flow on arc 4 would be altered. For path 2, the reliability is 1, so it can only go down before changing the current solution. When it reaches 0.95, the flow on arc 4 would be changed. For path 3, the performance of the example network would not change as long as the reliability of path 3 is between 0.0 and 0.1.

Just as one was interested in knowing the effects of single changes in the path reliabilities, one would be interested in obtaining similar information about the component capacities. Earlier, the bottlenecks were identified as arcs 1, 4, and 7. The effect of a unit increase in the capacity of these bottlenecks on the network can also be obtained from Appendix C.4. Increasing the capacity of arc 1 by one unit increases the objective function (network performance) by 0.9 unit, whereas increasing the capacity of arcs 4 or 7 by one unit could only increase the objective function by 0.05 unit. Thus, with all other capacities held at their original values, changing the capacity of arc 1 by one unit would most significantly affect the performance of the network.

To improve the performance of the network, the capacities of the arcs that formed the bottlenecks could be increased, but the improved network might again contain bottlenecks without reaching the optimal solution. Furthermore, it would be

difficult to determine whether the solution obtained were optimal or not. To avoid this inefficiency and uncertainty, the investment strategy model 1 or 2 were used to determine the optimal investment strategy that maximized the network performance.

The solution to Investment Strategy Model 1, shown in Table 16, is shown in Appendix C.6. Under the assumption that the cost of increasing the capacity by one unit and budget available are as follows: $c_1 = 50$, $c_2 = 60$, $c_3 = 20$, $c_4 = 40$, $c_5 = 50$, $c_6 = 40$, $c_7 = 70$, abd $B = 1000$, the investments were made in four arcs as described below:

No.	Arc	Arc Survivability	Capacity Increase
1	arc 1	1	10.91
2	arc 3	1	6.91
3	arc 4	1	3.00
4	arc 6	0.9	4.91

With this new investment strategy, path 1 carried 8.91 units of flow, and path 2 carried 7.00 units of flow. Path 3 was no longer used. The new performance level, the sum of expected path flows, was $(8.91)(0.9) + (7.00)(1) = 15.02$ units.

The solution to the Investment Strategy Model 2, shown in Table 17, is shown in Appendix C.7. The optimal investment strategy was to invest in the same components as before. But, because the capacities could only be improved in increments of 5 units, the amount of capacity increase in each of these components were slightly different as shown below:

No.	Arc	Arc Survivability	Capacity Increase
1	arc 1	1	10
2	arc 3	1	5
3	arc 4	1	5
4	arc 6	0.9	5

Again, only paths 1 and 2 carried the flow from source to sink. The new performance level was 13.3.

7.2 Case Study

In order to accomplish complete analysis of the three "realistic" networks, the following questions were answered for each network:

1) What is the performance level of the network?

2) Which paths in the network are significant?

3) Which components in the network are bottlenecks?

4) What is the optimal investment strategy that maximizes the network performance?

In addition to the above questions, a specific question, regarding the survival probability of the significant node, was answered.

7.2.1 Analysis of Network A The revised topology of Network A is shown in Appendix D.1. The input data, which describes this revised network, for the FORMULA program is shown in Appendix D.2.

Question 1: What is the performance level of the network?

Appendices D.4, D.5, and D.6 contain solutions to maximum flow, lower bound, and upper bound models, respectivity. The performance of the network A is summa-

rized in Figure 14. Under adverse conditions, when the components of the network were subject to failure, the minimum expected amount of information the network could handle was 167 units. On the other hand, the maximum expected amount of information it could handle was 5760 units. These lower and upper bound of expected maximum flow provided the range of expected network performance. The expected performance as determined by the Monte Carlo simulation [2] was 619 units. The absolute best performance that this network could have was 9600 units; it occured when the components were perfectly reliable with survival probability of 1 for all components.

Question 2: Which paths in the network are significant?

The listing of paths and path reliabilities in Appendix D.3 shows that the network A contained 63 paths from source to sink. Appendix D.5 contains one of the solutions to the lower bound of expected maximum flow model; there were alternate solutions. From Appendix D.5, only eight paths were used to carry the flow (information) from source to sink. These eight paths are identified below:

Number	Path	Path Reliability	Path Flow
1	path 1*: 4,14,19,16	0.03600	1200
2	path 4 : 2,14,19,16	0.01296	1200
3	path 24: 2,5,11,15,8,14,17,16	0.00035	1200
4	path 25: 3,11,15,6,14,19,16	0.00726	1200
5	path 36: 3,9,15,6,14,17,16	0.00296	1200
6	path 45: 1,12,15,6,14,17,16	0.00178	1200
7	path 54: 1,13,15,6,14,17,16	0.00593	1200
8	path 61: 1,14,19,16	0.07200	1200

* Path 1 consists of node 4, 14, 19, and 16.

[2]Simulation results were obtained from independent simulation analysis

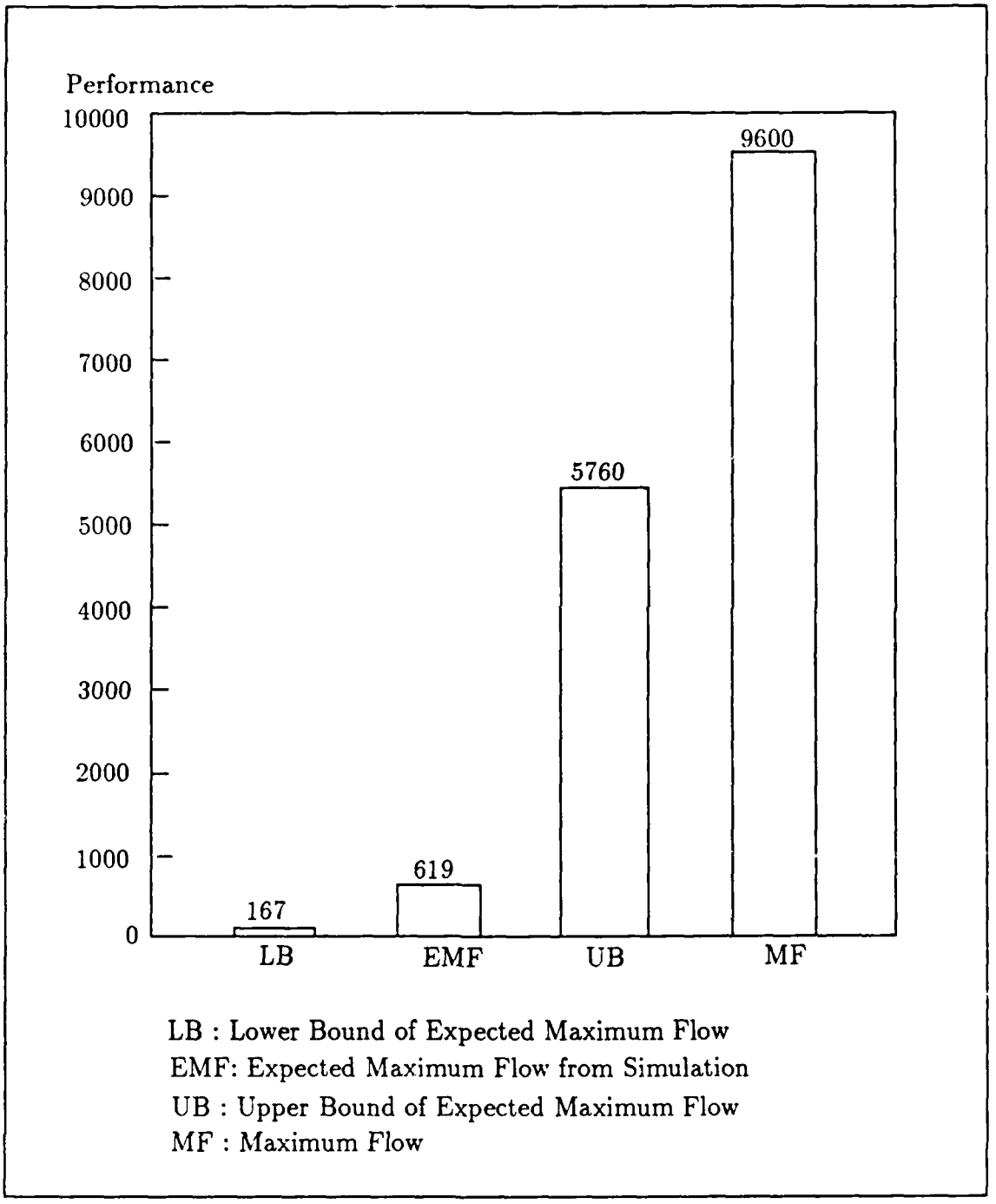


Figure 14. Performance Level of Network A

Although, all eight paths carried the same amount of flow, their expected path flow ranged from 0.42 to 86.4 units, because each path had a different reliability. Path 61 was most significant, since it had the highest expected path flow.

Question 3: Which components in the network are bottlenecks?

Since all nodes were noncapacitated, only arcs became the bottlenecks in the system. Deducing from the results shown in Appendix D.5, the bottlenecks are identified, with thicklines, in Figure 15. As suspected, all arcs located prior to node 11 were identified as bottlenecks.

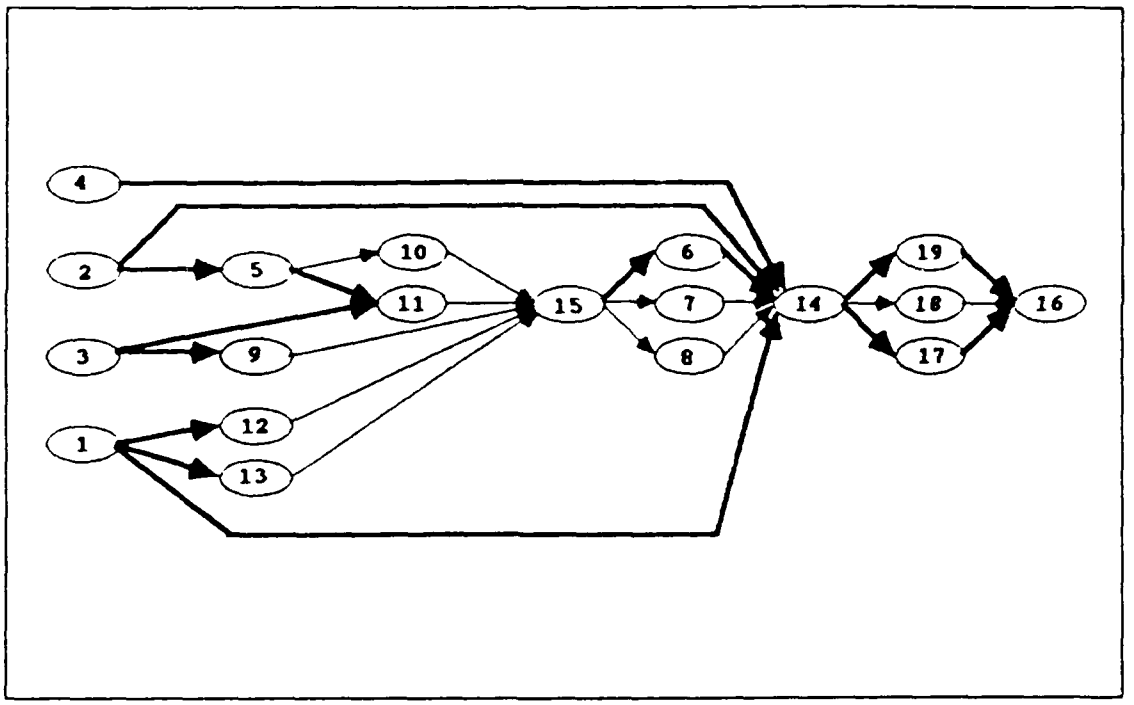


Figure 15. Bottleneck Arcs in Network A

Question 4: What is the optimal investment strategy that maximizes the network performance?

The improvement strategy would depend greatly on the investment cost in each component and total budget available for the investment. Three different cases were considered.

Case 1: Assume the cost of increasing one unit of capacity for each component, c_i, is \$100, and the total budget available for investment, B, is \$100,000. What is the optimal investment strategy?

This problem could be solved using the Investment Strategy Model 1. Here, $c_i = 100$ and $B = 100000$. The results of computer run for this model is shown in Appendix D.7. From Appendix D.7, all investments were made at arc 31 which starts at node 1 and terminates at node 14. It had a survival probability of 1 and an original capacity of 1200 units. Increasing the capacity of this arc by 1000 units increased network performance to 237.7.

With this investment strategy, the flows on the previous paths were altered. Two paths, path 23 and 27, were newly used to acommodate the flow. While the flow on paths 24, 25, and 27 dewindled, pathe 61, which contained an improved arc, carried an additional 1000 units of flow. The ten paths that carried the flow are described below:

Number	Path	Path Reliability	Path Flow
1	path 1 : 4,14,19,16	0.03600	1200
2	path 4 : 2,14,19,16	0.01296	1200
3	path 23: 2,5,11,15,8,14,18,16	0.00026	1000
4	path 24: 2,5,11,15,8,14,17,16	0.00035	200
5	path 25: 3,11,15,6,14,19,16	0.00726	200
6	path 27: 3,11,15,6,14,17,16	0.00593	1000
7	path 36: 3,9,15,6,14,17,16	0.00296	1200
8	path 45: 1,12,15,6,14,17,16	0.00178	1200
9	path 54: 1,13,15,6,14,17,16	0.00593	1200
10	path 61: 1,14,19,16	0.07200	1200

Case 2: Again, assume $c_i = 100$ and B = 100000. Now, there is a limit on how much capacity can be increased on each component when the investment is made. This limit, y_i, is 100 units for all components. What is the optimal investment strategy?

This problem could again be solved using the Investment Strategy Model 1. But, the following new constraint had to be added to limit the capacity increase to 100 units:

$$d_i \leq 100 \qquad\qquad i = 1, 2, ..., n.$$

The results of the computer run for this model is shown in Appendix D.8. With the above restriction, the investments were distributed among 11 components as shown below:

No.	Arc	Start Node	Term Node	Arc Survivability	Capacity Increase
1	arc 24	4	14	1.0	100
2	arc 25	2	14	0.6	100
3	arc 27	3	11	1.0	100
4	arc 28	3	9	1.0	100
5	arc 29	1	12	1.0	100
6	arc 30	1	13	1.0	100
7	arc 31	1	14	1.0	100
8	arc 39	15	6	0.3	50
9	arc 42	6	14	0.6	50
10	arc 45	14	19	0.6	100
11	arc 48	19	16	0.3	100

A majority of investment went to the arcs with high survival probabilities. After investing in these components, the performance level increased to 180.4 unit. The flows on ten paths described in Case 1 were rerouted, and path 51 was newly used to carry some of the flow. See Appendix D.8 for the details of flow routing.

Case 3: Assume c_i = 100 and B = 100000. There is no limit on how much investment can be made on each component, but the capacity can only be increased in increment of predetermined amount, b_i. Thus, each investment causes a step-wise improvement in the component capacities. Let b_i be 100 for all components. What is the optimal investment strategy?

The optimal investment strategy, under conditions specified in the problem statement, could be obtained by using the Investment Strategy Model 2. Here, $c_i = 100$, $B = 100000$, and $b_i = 100$. The decision variables, g_i, could only have an integer solution. The results of computer run for this model is shown in Appendix D.9. The optimal investment strategy for this case turned out to be the same as Case 1. All investment went to Arc 31, increasing the performance level to 237.7.

Question 5: What is the effect of the survival probability of node 14 on the performance of the network?[3]

Question 4 mainly studied the effect of capacity increase on the network performance. The effect of component's survival probabilities could also be studied using same general models. The method used here was rather "brute force" way of determining the effect of the component's sruvival probabilities, but it served its purpose. The method involved implementing the new survival probabilities when building the models, then determining the new network performance level.

Since all 63 paths, in Network A, went through node 14, the strategic importance of node 14 could not be overemphasized; but how exactly did the survivability of this node affect the overall network performance? Figure 16 summarizes the effect of survival probability of node 14 on the performance of the network.

Of course, when the survivability of this node was 0, the network performance level was 0, since all paths were broken. As the survivability increased, the performance level increased linearly. The performance reached its maximum level of 334 when the survivability of node 14 reached 1.

7.2.2 Analysis of Network B Since the procedure of analyzing Network B was same as that of Network A, detailed explanations that seem redundent are omitted.

[3]This question demonstrates one way to study the effect of changing survival probabilities of the components.

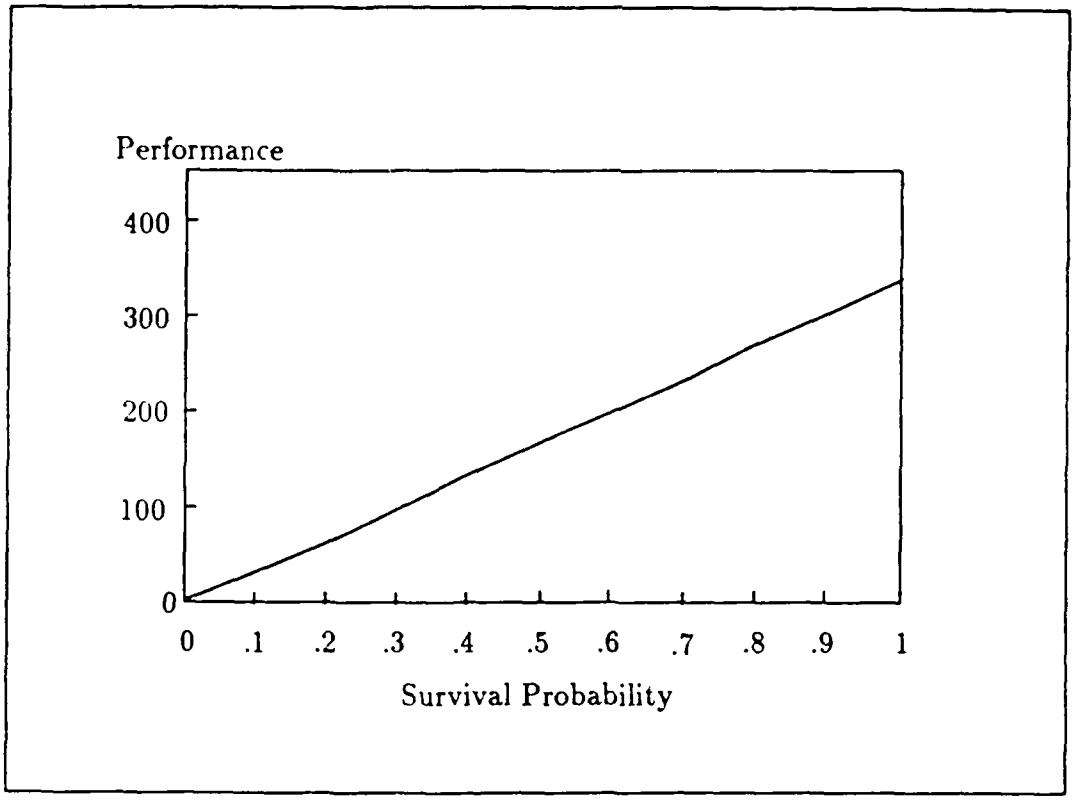


Figure 16. Effect of Survival Probability of Node 14

Also the results of the computer runs are omitted. The models were developed based on the revised Network B shown in Appendix E.

Question 1: What is the performance level of the network?

The performance level is shown in Figure 17. The minimum expected amount of information the network could handle was 344 units; the maximum expected amount of information it could handle was 2040 units. The expected performance was 354, while the absolute best performance that this network could have was 3900.

Question 2: Which paths in the network are significant?

The network B contained 187 paths from source to sink. There were alternate solutions to the lower bound of expected maximum flow model; one of the solutions

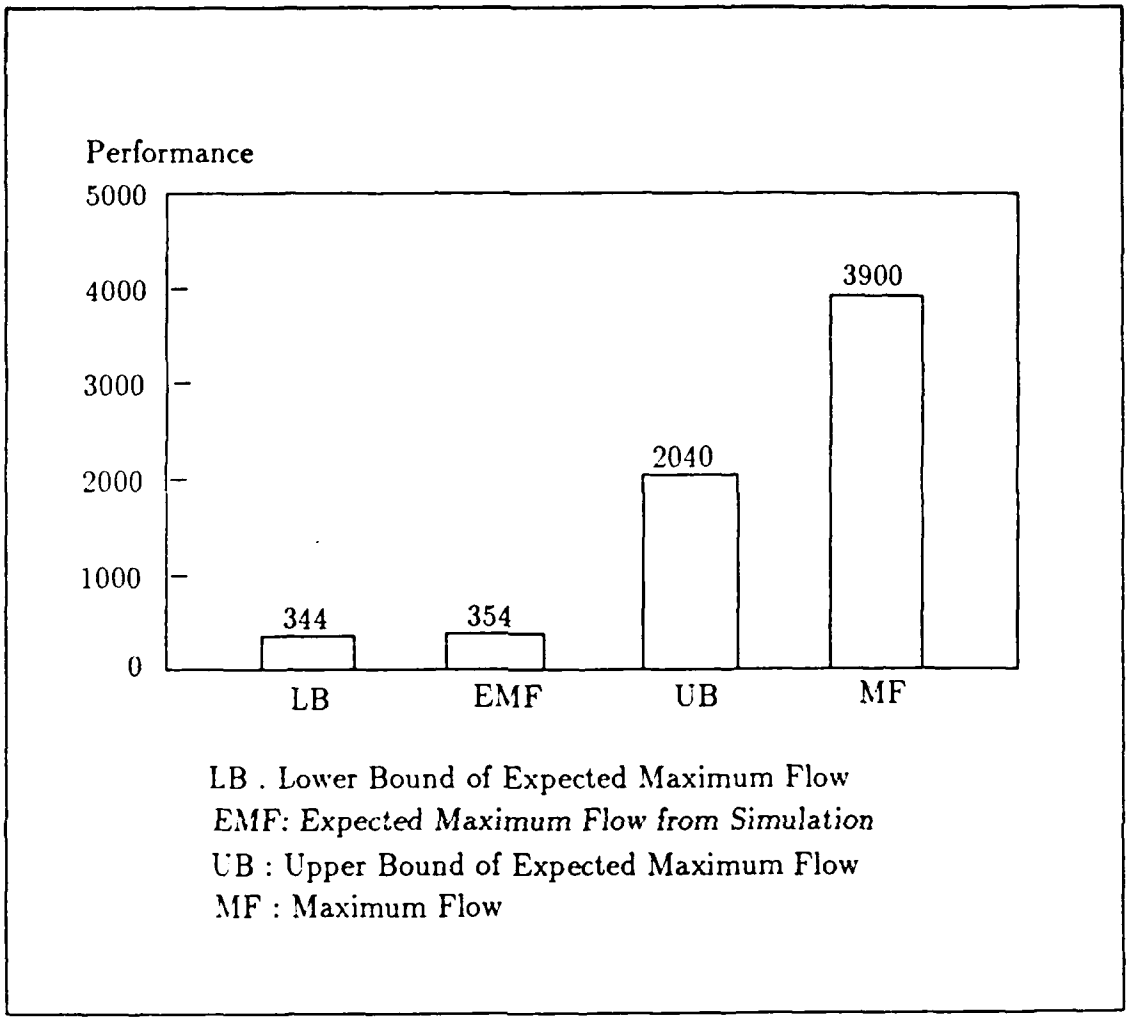


Figure 17. Performance Level of Network B

indicated that 22 paths were used to carry the flow form source to sink. These 22 paths are identified below:

Number	Path	Path Reliability	Path Flow
1	path 4 : 1,7,10,12,20	0.02744	150
2	path 28 : 1,8,10,12,20	0.09604	200
3	path 52 : 2,8,10,12,20	0.01286	750
4	path 74 : 2,9,10,11,18	0.00001	75
5	path 75 : 2,9,10,11,19	0.00001	675
6	path 97 : 3,7,10,11,17	0.00009	100

7	path 100: 3,7,10,20	0.00118	100
8	path 121: 3,9,10,11,17	0.000001	350
9	path 161: 4,16,23	0.03360	50
10	path 162: 4,16,24	0.03360	75
11	path 163: 4,16,25	0.03360	75
12	path 165: 4,24	0.50000	600
13	path 176: 6,16,17	0.00084	75
14	path 177: 6,16,18	0.00084	75
15	path 178: 6,16,19	0.00084	75
16	path 179: 6,16,20	0.00084	75
17	path 180: 6,16,21	0.00084	75
18	path 181: 6,16,22	0.00084	75
19	path 182: 6,16,23	0.00084	25
20	path 185: 6,16,26	0.00084	75
21	path 186: 6,25	0.02400	75
22	path 187: 6,26	0.02400	75

Most of the 22 paths had low reliabilities. Path 165, which consisted of only one arc, had survival probability of 0.5. Because of this high survivability, this path would be most significant. Most investments would go to this path to improve network performance.

Question 3: Which components in the network are bottlenecks?

The bottlenecks are identified, with thick lines, in Figure 18. In this network, the bottlenecks were located near the sink nodes. The flows on all arcs starting at node 16 reached their maximum capacities.

Question 4: What is the optimal investment strategy that maximizes the network performance?

To study the investment strategies for Network B, two different cases were considered.

Case 1: Let $c_i = 100$ and $B = 100000$. What is the optimal investment strategy?

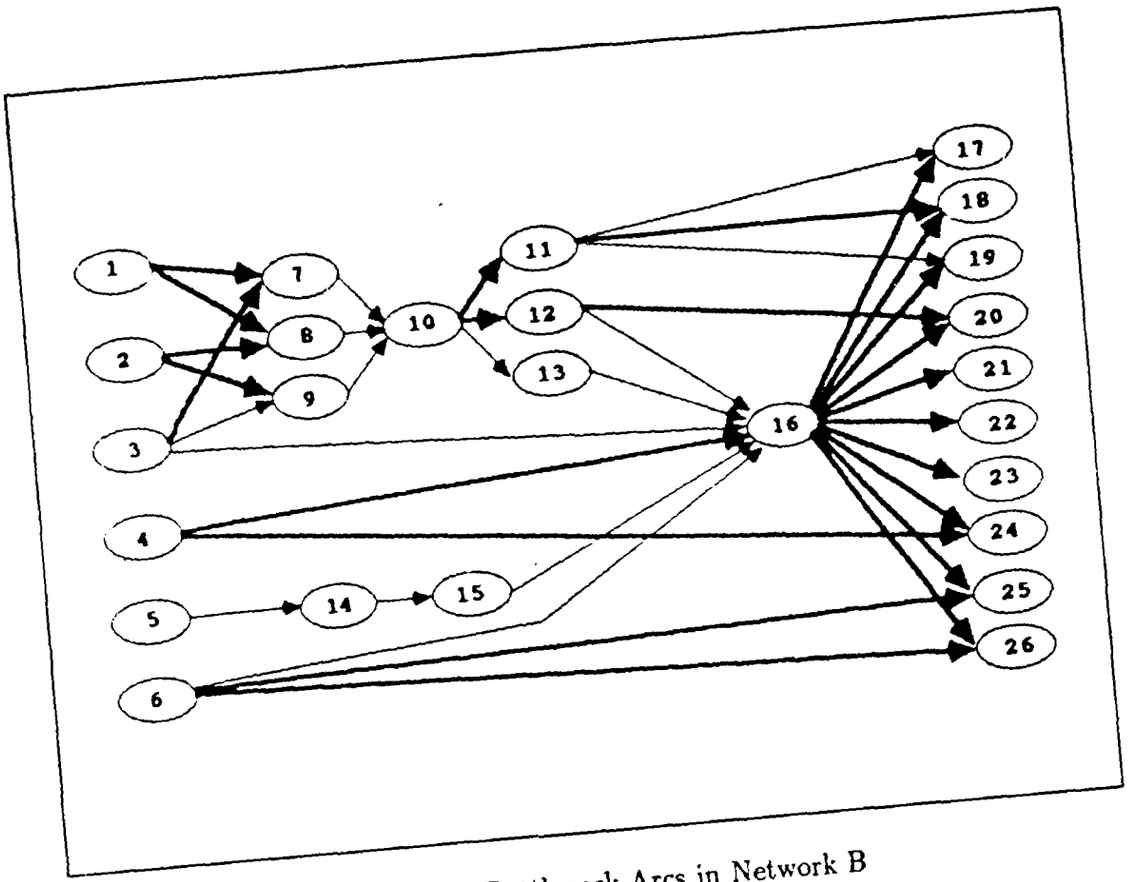


Figure 18. Bottleneck Arcs in Network B

This problem was solved using the Investment Strategy Model 1. As expected, all investments were made on arc 53 which was a path – path 165 – by itself starting at node 4 and terminating at node 24. Increasing the capacity of this arc by 1000 units increased network performance to 843.9. With this investment strategy, the flows on the previous paths remained same, except the flow on path 165, which contained arc 53, increased to 1600 units utilizing all 1000 units of added capacity.

Case 2: Let $c_i = 100$ and $B = 100000$. Here, the investment can be made only once on each component by predetermined amount of capacity increase, b_i. Let b_i be 500 for all components. What is the optimal investment strategy?

This problem was solved using the Investment Strategy Model 2. For this problem, the investment decision variables, g_i, were incorporated as zero-one variables. Zero corresponds to the decision not to invest, and one corresponds to the

decision to invest in the component. Thus, zero-one decision variables allowed only one investment to be made by predetermined amount on each component.

The optimal investment strategy was to invest 500 units of capacity on arc 34 and arc 53 each. Arc 34 starts at node 1 and terminates at node 8, and arc 53 starts at node 4 and terminates at node 24. By making these investments, network performance level increased to 636.8. Also, the flow routing has been changed. Paths 74 and 75 were no longer used; instead, three new paths were used to carry some of the flow. These three new paths are identified below:

Number	Path	Path Reliability	Path Flow
1	path 51: 2,8,10,11,19	0.00101	150
2	path 73: 2,9,10,11,17	0.00001	750
3	path 98: 3,7,10,11,18	0.00092	75

Question 5: What is the effect of the survival probability of node 16 on the performance of the network when the investment is made?

By implementing new survival probabilities into the investment strategy models, the joint effect of both survivabilities and capacities on the optimal investment strategy could be studied. Reviewing the topology of Network B, a majority of the paths went through the node 16, but there were a number of paths which did not go through the node 16. Thus, even when the node 16 was down, the network could still carry the flow from source to sink. Figure 19 summarizes the effect of survival probability of node 16 on the performance of the network when the investment is made. The performance level was 837 when the survivability of the node 16 was 0. As the survivability of this node increased, the performance level went up slightly, but not significantly. All investment went to arc 53 as before. Although, many paths go through node 16, investing in this node to increase its survivability would bring very little return.

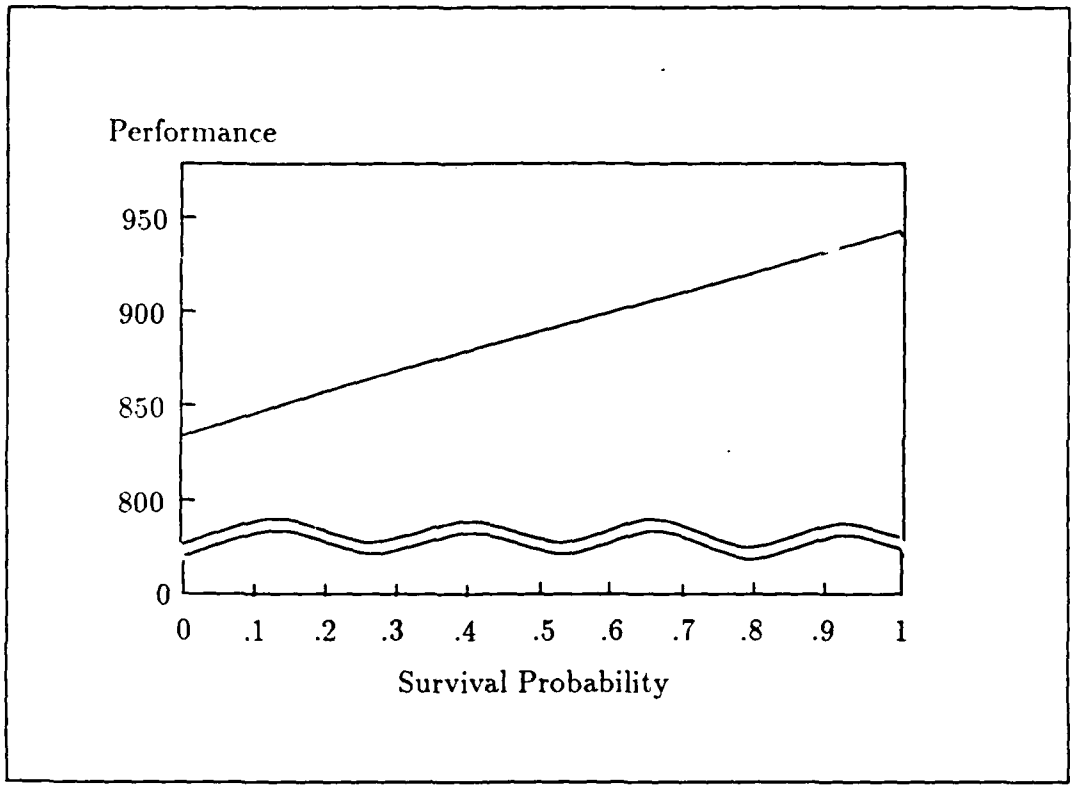


Figure 19. Effect of Survival Probability of Node 16

7.2.3 Analysis of Network C The revised Network C is shown in Appendix F.

Question 1: What is the performance level of the network?

The performance level is shown in Figure 20. The minimum expected amount of information the network could handle was 855 units; the maximum expected amount of information it could handle was 4290. The expected performance as determined by the Monte Carlo simulation was 1169. The maximum flow through this network when the network was 100% reliable was 6300.

Question 2: Which paths in the network are significant?

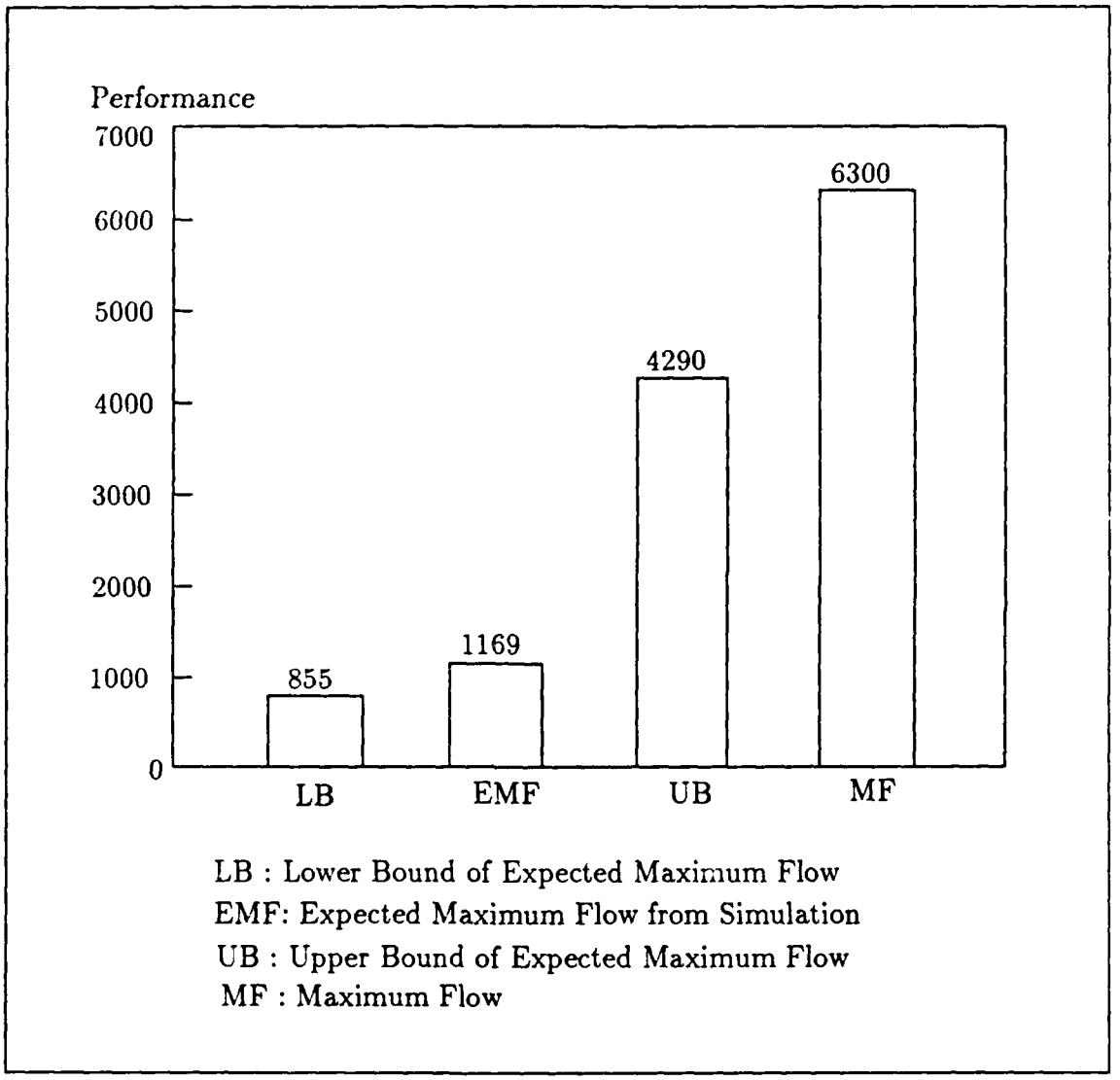


Figure 20. Performance Level of Network C

Network C contained 198 paths from source to sink. The solution indicated that only eight paths were used to carry the flow from source to sink. These eight paths are identified below:

Number	Path	Path Reliability	Path Flow
1	path 33 : 1,11,39	0.10584	1200
2	path 62 : 2,11,12,20,35	0.30870	1200
3	path 82 : 3,7,10,11,12,14,29	0.06915	300
4	path 128: 4,11,12,20,35	0.21168	1200
5	path 146: 5,8,11,12,14,27	0.03001	300
6	path 160: 5,8,11,12,19,39	0.03086	900
7	path 181: 6,9,11,12,14,29	0.04064	900
8	path 193: 6,9,11,12,19,39	0.03292	300

Question 3: Which components in the network are bottlenecks?

The bottlenecks are identified, with thick lines, in Figure 21. All arcs located prior to node 11 were bottlenecks.

Question 4: What is the optimal investment strategy that maximizes the network performance?

Again, two different cases were considered to study the investment strategies.

Case 1: Assume the cost of increasing one unit of capacity is one tenth of component's capacity. For example, an arc that has 1200 units of capacity costs $120 to increase the capacity of that arc by one unit. The total investment budget available is $1,000,000. What is the optimal investment strategy?

The solution obtained from solving the Investment Strategy Model 1 indicated that arcs 47 and 56 should be invested in. Arc 47 starts at node 2 and terminates at node 11, and arc 56 starts at node 11 and terminates at node 23. To obtain the

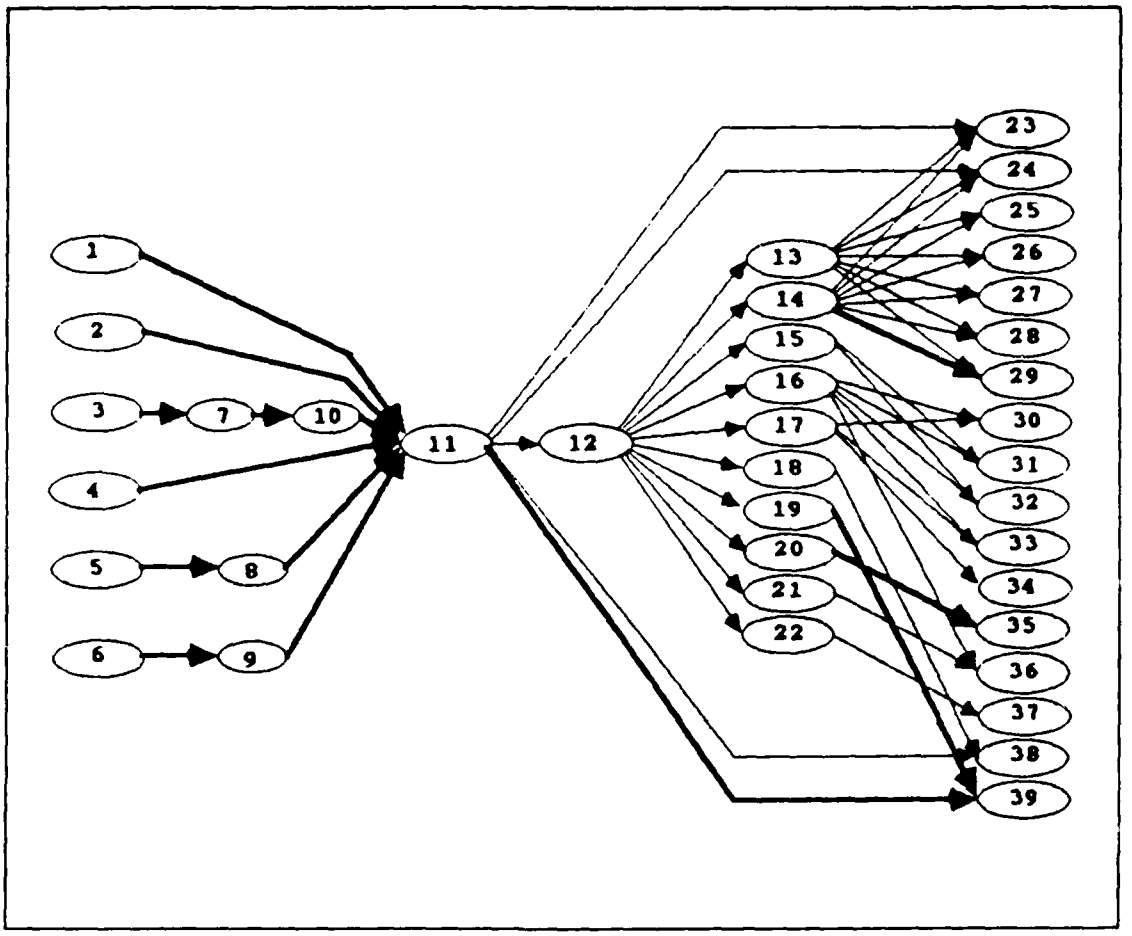


Figure 21. Bottleneck Arcs in Network C

maximum performance level, the capacity of arc 47 should be increased by 8188.3 units and arc 56 by 2337.3 units. Note that arc 56 was not a bottleneck, but improving the capacity of this arc was necessary to obtain the highest return on the investment. With this investment strategy, the performance level would increase to 2275.5. The flows on the previous paths were altered considerably; paths containing improved arcs carried a majority of flows.

Case 2: The c_i and B are assumed same as Case 1, but the capacity can only be increased in increment of predetermined amount,b_i, of 300 units for all components. What is the optimal investment strategy?

The optimal investment strategy obtained from solving the Investment Strategy Model 2 was very similar to Case 1. All investments went to arcs 47 and 56 as before, and improved network performance level was 2272. There was $1,000 left in the budget. This was because $1,000 was not enough to improve any component by 300 units.

Question 5: What is the effect of the survival probability of node 11 on the performance of the network?

Just like in Network A, all paths went through node 11 in Network C. The survivability of this node would be crucial to network performance. Figure 22 summarizes the effect of survival probability of node 11 on the performance of the network. The survivability of this node turned out to be very significant to the network performance. Increasing the survivability of this node by 0.1 increased the network performance by approximately 122 units. When this node became perfectly reliable, the performance reached its maximum level of 1222.

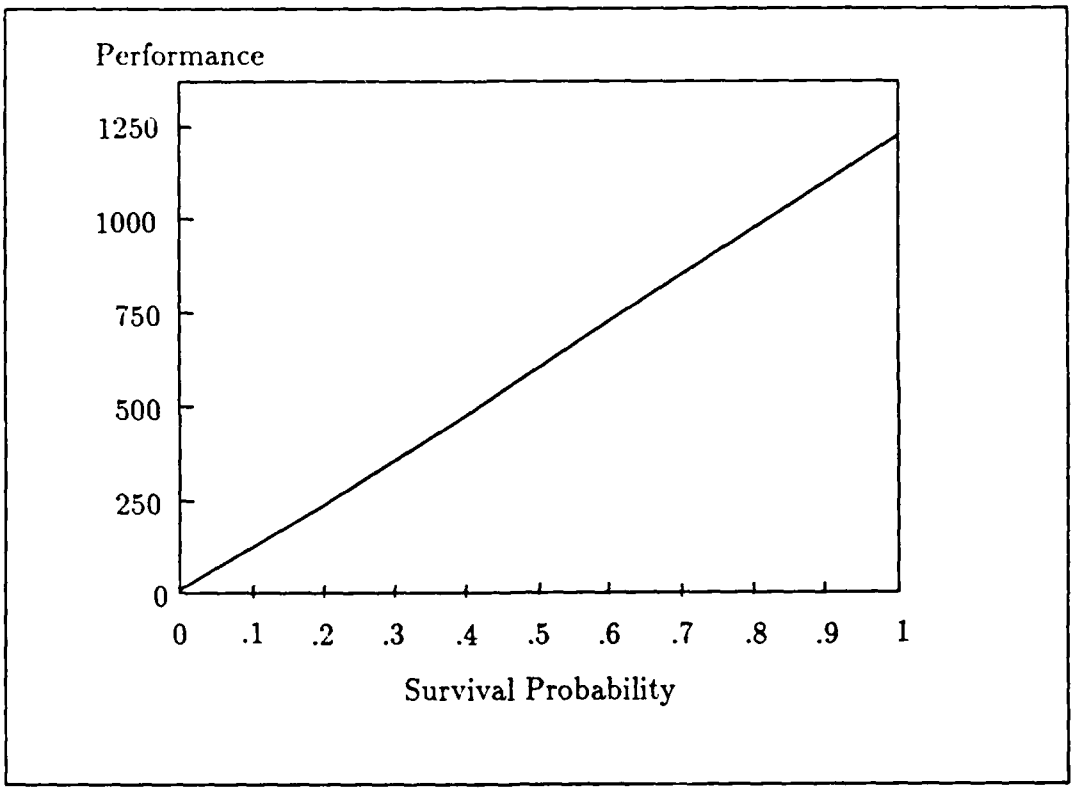


Figure 22. Effect of Survival Probability of Node 11

VIII. Conclusions and Recommendations

This chapter presents the summary of the research project, its conclusions, and recommendations for future research.

8.1 Summary

The goal of this research was to develop an analytical methodology to assess the performance of a stochastic communication network and to determine the optimal investment strategy to improve the performance of that network. The performance could be assessed either by measuring the expected maximum flow or the reliability of the network. In this research, the expected maximum flow was chosen to use as a measure of network performance. However, since calculating the exact value of expected maximum flow was intractable, the bounds of expected maximum flow were investigated. Specifically, the lower bound was used as a measure of stochastic network performance.

Linear programming models were developed to measure the maximum flow through the network, the lower bound of expected maximum flow, and the upper bound of expected maximum flow. The maximum flow provided the performance level of a perfectly reliable network; the bounds provided the range of performance level of a stochastic network. These performance models were developed based on the arc-path incidence matrix.

The improvement was made by increasing the capacity of the components. Two mathematical models were developed to determine the optimal investment strategy to maximize the lower bound. The first model was a linear programming model. The solution obtained from this model determined which components should be improved and by how much. The second model was a mixed integer programming model. This model required the user to supply the fixed amount of capacity increase in each component, then it determined which components should be improved in order

to maximize the lower bound. Of course, if the components were not capacitated, then there was no need to invest. Both of these models were based on the arc-path incidence matrix.

The performance models and investment models were easy to understand in theory; but developing these models in practice requires a fair amount of effort. The complexity was due to the fact that the models were based on the arc-path incidence matrix. The formulations of these models required finding all paths in the network from source to sink, calculating all paths reliabilities, and finding all paths that went through each arc. To minimize the effort in developing these models, a Prolog program, FORMULA, was developed. This program not only found paths and calculated reliabilities, it generated all performance and investment strategy models. The output of this program served as a direct input file to LP/MIP-83 linear and mixed integer programming system.

Using the general mathematical models developed, one example network and three "realistic" communication networks, namely Network A, B, and C, were analyzed. For the example network, the sensitivity analysis was performed to investigate the effects of single changes of path reliabilities or component capacities on the network performance.

The comparison of network performance levels of the three "realistic" networks were rather trivial, because each network had a different topology and characteristics. However, among the three networks, the Network C was most survivable in terms of the amount of information received at the sink nodes. Network C's lower-bound-performance level was 855 units, whereas, Network B was 344, and Network A was 167 units. Both Network A and C had one central node through which all paths from source to sink went. The survivability of these central nodes were crucial to the performance of the network. The central node (node 11) of Network C was even more important, because for every 0.1 increase in the survivability of this node, the performance level increased by 122 units. For Network A, increasing the survivability

of the central node (node 14) by 0.1 only increased the performance level by 33 units. In general, the analytical models precisely determined the current performance level, identified important paths, identified the bottlenecks, and determined optimal investment strategy to improve network performance.

8.2 Conclusions

There were several conclusions drawn from this research effort. First, the analytical models were very helpful to quickly determine the current performance levels of the network and to identify bottlenecks in the system. Using the LP/MIP-83, the solution for each model was found in less than 40 seconds. The models also identified important paths which should be protected during adverse conditions to keep the performance level at its current status. The greatest advantage of the analytical models was its ability to determine the optimal investment strategy to maximize the network performance level. The investment strategy models identified the exact location and amount of investments as well as new performance level.

Second, the use of the mathematical programming models provided a room for the sensitivity analysis. Through sensitivity analysis, the exact effect of a single component characteristic on the network performance could be determined. In designing a new communication network or upgrading the capability of an existing network, knowing the effect not only saves time, but prevents costly mistakes beforehand of improving the wrong components.

Third, knowing the bounds of expected performance might be more useful than just knowing the exact value. The bounds showed the range of expected performance. Because the survival probabilities of components were always approximate, the exact value of expected maximum flow might not be significant.

Fourth, the size of the model grew linearly with the number of paths in the network from source to sink. Each path flow was represented as a variable in the objective function of the model. If the network contained more than approximately

1200 paths, the LP/MIP-83 could not handle the problem, although the program, FORMULA, was well capable of generating large models. The largest network, Network C, analyzed in this research used approximately 23% of the total capacity of the LP/MIP-83.

Fifth, artificial intelligence may play an important role in operations research. In this research, a depth-first search technique was implemented using the Prolog to find all paths in the network. The branch and bound algorithm could also be implemented using artificial intelligence techniques. Perhaps, calculating the exact value of expected maximum flow could be accomplished in real time using the artificial intelligence in near future.

8.3 Recommendations

The following are recommendations for future research:

First, study the relationship between the lower bound of expected maximum flow and the exact value of expected maximum flow. In this research, attempts were made to determine this relationship, but so far they were unsuccessful.

Second, Modify the program, FORMULA, so that the output can be used as an input to other mathematical programming packages such as SAS and MINOS. Although, FORMULA successfully generated the model for the network that contained approximately 2200 variables and 112 constraint equations, the LP/MIP-83 could not handle this size of a problem. The maximum capacity of LP/MIP-83 was about 1200 variables.

Third, instead of investigating the bounds, investigate the algorithm [22, 30] that determines approximate value of expected maximum flow by enumerating the most probable states of the network.

Bibliography

1. Aneja, Y. P. and K. P. K. Nair. "Maximal Expected Flow in a Network Subject to Arc Failure," *Networks, 10*: 45-57 (1980).
2. Arity Corporation. *The Arity/Prolog Language Reference Manual.* Concord, Masschusetts, 1988.
3. Arity Corporation. *Using the Arity/Prolog Compiler and Interpreter.* Concord, Masschusetts, 1987.
4. Ball, Michael O. "Complexity of Network Reliability Computations," *Networks, 10*: 153-165 (1980).
5. Ball, Michael O. "Computing Network Reliability," *Operations Research, 27*: 822-838 (July-August, 1979).
6. Beasley, J. E. "Supercomputers and OR", *Journal of Operations Research Society, 38* 1085-1089 (1987).
7. Bharath, Ramachandran. "Logic Programming: A Tool for MS/OR?," *Interface, 16*: 80-91 (1986).
8. Brako, Ivan. *PROLOG Programming for Artificial Intelligence.* Menlo Park, California: Addison-Wesley Publishing Company, 1987.
9. Brooks Rodney A. *Programming in Common LISP.* New York: John Wiley and Sons, 1985.
10. Carey, Malachy., and Chris Hendrickson. "Bounds on Expected Performance of Networks with Links Subject to Failure," *Networks, 14*: 439-456 (1984).
11. Chan, Yupo. *Survival Communication Network.* Thesis topic description. Department of Operations Science, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, February, 1988.
12. Chan, Yupo. "Transportation Network Investment Problem - A Synthesis of Tree-Search Algorithms," *Economic Factors in the Provision of Transportation Services* . Transportation Research Record 1074. Transportation Research Board, Washington, D.C., (1986).
13. Chvatal, Vasek. *Linear Programming.* New York: W. H. Freeman and Company, 1983.
14. Dowsland, William B. "Microcomputer Software and Hardware Considerations for OR," *Journal of Operations Research Society, 38*: 87-93 (1987).
15. Edmonds, J. and R. M. Karp. "Theoretical Improvements in Logorithmic Efficiency for Network Flow Problems," *Jour. ACM, 19*: 248-264 (April, 1972).
16. Evans, J. R. "Maximum Flow in Probabilistic Graphs - The Discrete Case," *Networks, 6*: 161-183 (1976).

17. Ford, L. R. and D. R. Fulkerson. *Flows in Networks.* Princeton University, Princeton, NJ 1962.

18. Hansler, E., et al. "Exact Calculation of Computer Network Reliability," *AFIPS Conference Proceedings, 41*: 49-54 (1972).

19. Hillier, Frederick S. and Gerald J. Lieverman. *Introduction to Operations Research (Fourth Edition).* Oakland: Holden-Day, Inc., 1986.

20. Howard, Frank and Ivan T. Frisch. "Analysis and Design of Survival Networks," *IEEE Transactions on Communication Technology, COM-18*: 501-519 (October, 1970).

21. Hu, David. *Programmer's Reference Guide to Expert System.* Indianapolis: Howard W. SAMS and Company, 1987.

22. Lam, Y. F. and V. O. Li. "An Improved Algorithm for Performance Analysis of Networks with Unreliable Components," *IEEE Transactions on Communication, COM-34*: 496-497 (May, 1986).

23. Lin, P. M., et al. *Some New Algorithms for Deterministic and Probabilistic Communication Networks*, June 1974. (AD-785 119)

24. Mandl, Christoph. *Applied Network Optimization.* New York: Academic Press, 1979.

25. Marcus, Claudia. *Prolog Programming: Application for Database Systems, Expert Systems and Natural Language Systems.* Reading, Massechusetts: Addison-Wesley Publishing Company, Inc., 1986.

26. Onage K. "Bounds on the Average Terminal Capacity of Probabilistic Nets," *IEEE Transactions on Information Theory*: 766-768 (September, 1968).

27. Ove, Frank and Gaul, Wolfgang. "On Reliability in Stochastic Graphics," *Networks, 12*: 119-126 (1982).

28. Rich, Elaine. *Artificial Intelligence.* New York: McGraw-Hill Book Company, 1983.

29. Sancho, N. G. F. "On the Maximum Expected Flow in a Network," *Journal of the Operationsl Research Society, 39*: 481-485 (May, 1988).

30. Shier D. R. "A New Algorithm for Performance Analysis of Communication Systems," *IEEE Transactions on Communications, 36*: 516-519 (April, 1988).

31. Sunset Software. *LP/MIP 83: A Professional Linear and Mixed Integer Programming System.* San Marino, California, 1985.

32. Van, Slyke R. and H. Frank. "Network Reliability Analysis: Part I," *Networks, 1*: 279-290 (1972).

33. Wallace, Stein W. "Investing in Arcs in a Network to Maximize the Expected Max Flow," *Networks, 17*: 87-103 (Spring, 1987).

34. Wilkov, Robert S. "Analysis and Design of Reliable Computer Networks," *IEEE Transactions on Communications, COM-20*: 660-678 (June, 1972).

Appendix A. *FORMULA Program*

A.1 FORMULA.ARI: Main Program

```
/*============================================================*/
/*                                                            */
/*                    FORMULA   Version 1.0                   */
/*                                                            */
/*============================================================*/
/*                                                            */
/* This program does the following six tasks:                 */
/*                                                            */
/*  1) Finds all paths in the network from source (s) to      */
/*     sink (t) and calculates all path reliabilities.        */
/*  2) Generates the formulation of the Maximum Flow through  */
/*     the network.                                           */
/*  3) Generates the formulation of the Lower Bound of the    */
/*     Expected Maximum Flow.                                 */
/*  4) Generates the formulation of the Upper Bound of the    */
/*     Expected Maximum Flow.                                 */
/*  5) Generates the formulation of the investment strategy   */
/*     model 1.                                               */
/*  6) Generates the formulation of the investment strategy   */
/*     model 2.                                               */
/*                                                            */
/* The five mathematical programming models (2 to 6) are      */
/* developed based on arc-path incidence matrix built from    */
/* the description of the network in the input file.          */
/* The network described in the input file must contain a     */
/* single source node, named 's', and sink node, named 't'.   */
/* All capacitated or stochastic nodes must be represented    */
/* as a dummy arc with two nodes. Refer to FORMULA user's     */
/* manual for details on how to prepare the input file.       */
/*                                                            */
/* The formulations generated from this program is in the     */
/* same format as the input data file of LP/MIP 83 mathe-     */
/* matical programming package. Thus the ouput from this      */
/* program can be used as an input to LP/MIP 83 for further   */
/* analysis.                                                  */
/*                                                            */
/*------------------------------------------------------------*/
/*                                                            */
/*    PROGRAMMER: Capt Eugene Yim                             */
/*    DATE: 14 November, 1988                                 */
/*    FILENAME: FORMULA.ARI                                   */
/*                                                            */
/* This program was written in Prolog language using the      */
/* Arity/Prolog Version 5.0.                                  */
```

```
/*                                                              */
/*==============================================================*/

/* Start the program by typing 'go.' */

 go :-
     fileerrors(_,off),      % Turn off system file error message.
     windows,                % Call windows to display the
                             % introduction screen.
     open_input_datafile,    % Get the name of input file and
                             % consult it.
     start_program.
 go.

 start_program :-
     get_selection_number(Selection),
                             % Ask user what need to be done.
   ( Selection = 7,          % If '7'is selected, exit.
     clear_windows,!
   ;                         % Otherwise,
     nl,                     % Where should output be sent ?
     get_where_to_send_output(Where),
     execute_request(Selection,Where)   % Execute the request.
   ).


 execute_request(Selection,Where) :-
    ( Selection = 1,
      search_paths(Where),!
    ;
      Selection >= 2,
      Selection =< 4,
      performance_formulations(Selection,Where)
    ;
      Selection >= 5,
      Selection =< 6,
      investment_formulations(Selection,Where)
    ),
    get_run_again_reply(Reply),     % Run the program again ?
    cls,                            % Clear screen.
    ( Where = 1,                    % If the output was sent to
      true, !                       % the screen, do nothing.
    ;                               % Otherwise,
      exit_popup                    % delete popup window 'done'.
    ),
      want_more(Reply).

 want_more(Reply) :-
    ( Reply = 121,            % Reply is 'y' (ASCII 121),
      removeallh(matrix),     % delete hash table 'matrix', and
      start_program,!         % run program again.
```

```
    ;
     clear_windows,          % Otherwise, exit.
      nl
    ),!.


/*************************************************************************/
/* 'search_path' initiates search to find all paths in the network */
/* and calculates path reliabilities.                                    */
/*************************************************************************/

 search_paths(Where) :-
    ctr_set(1,1),        % Initialize counter one to 1
                         % to keep track of path number.
  ( Where = 1,           % When Where = 1, output is displayed
                         % on the monitor screen.
    continue_find_paths

  ;                      % 'Executing' message is displayed on
                         % the screen while the output is being
    executing_message,   % sent to the 'output1.lp' file.

    stdout('output1.lp',continue_find_paths),

    exit_popup,          % Delete popup window 'executing'.
    done_message(1)      % Display 'done' message.
  ).


 continue_find_paths :-
    nl,
    write('*************************************************************'),
    nl,
    write('*  Following is a list of all paths from "s" to "t"    *'),
    nl,
    write('*  of the network described in the input data file.    *'),
    nl,
    write('*************************************************************'),
    nl,nl,nl,
    find_paths(s,t),
    nl,nl,
    write('* ------ end ------ *'),
    nl,nl,nl,nl.

/*-----------------------------------------------------------------------*/

 find_paths(Start,Goal) :-        % Find all paths using depth-first
                                  % search method.
    depth_first([Start],Goal,Path),
    find_reliability(Path,Rel),   % Calculates the reliability of
                                  % a path.
```

```
   ctr_inc(1,Pnbr),                % Get current path number and
                                   % increment counter one by 1.
   display_outputs(Path,Pnbr,Rel),
   fail.                           % Go on to find next path until
                                   % there isn't any to be searched.
find_paths(_,_).


depth_first(Path,Goal,Path) :-     % Path is found if it satisfies
   satisfies(Path,Goal).           % Goal.

depth_first([X|Rest],Goal,Path) :-
   arc(X,Y),                       % Get next arc.
   not member(Y,[X|Rest]),         % Prevents cycles.
   depth_first([Y,X|Rest],Goal,Path).  % Recursive call.

satisfies([Goal|_],Goal).          % A path is found if the head of
                                   % a list describing Path matches
                                   % with Goal (t).

member(X,[X|Tail]).
member(X,[Head|Tail]) :-
   member(X,Tail).


find_reliability([],1) :- !.       % Reliability of an empty list is 1.
find_reliability([Arc|Rest],Rel) :-
   find_reliability(Rest,RelRest),  % Calculate Rel recursively.
   prob(Arc,Pb),                    % Get survival probability of Arc.
   Rel is Pb * RelRest.

/*-------------------------------------------------------------*/

display_outputs(Path,Pnbr,Rel) :-
   print_path(Path,Pnbr),              % Print path and
   print_reliability(Rel).             % reliability.


print_path(Path,Pnbr) :-
   write('  Path '),
   write(Pnbr),
   write(': '),
   write_reverse(Path).


print_reliability(Rel) :-
   nl,
   write('  Reliability: '),
   write(Rel), nl, nl.
```

```
write_reverse([]) :- !.              % Prints path from 's' to 't'.
write_reverse([Arc|Rest]) :-
   write_reverse(Rest),
   write(Arc), write(' ').


/*********************************************************************/
/* 'performance_formulations' generates the formulations of maximum */
/* flow, lower bound of expected maximum flow, and upper bound of   */
/* expected  maximum flow.                                          */
/*********************************************************************/


performance_formulations(Selection,Where) :-

  ( Where = 1,       % Display output on the screen.
    find_formulations(Selection)
  ;
    executing_message,              % Display output sending message.
    ( Selection = 2,
      stdout('output2.lp',find_formulations(Selection))
    ;
      Selection = 3,
      stdout('output3.lp',find_formulations(Selection))
    ;
      stdout('output4.lp',find_formulations(Selection))
    ),
    exit_popup,                     % Delete popup window 'executing'.
    done_message(Selection)         % Display 'done' message.
  ).

find_formulations(Selection) :-
    title(Selection),
    objective(Selection),
    constraints(Selection),
    nl,nl,
    write('* ------ end ------ *'),
    nl,nl,nl,nl.


/*-------------------------------------------------------------------*/

title(Selection) :-                 % Print title of the formulation.
   write(' ..Title'),
   nl,nl,
   ( Selection = 2,
     write(' Maximum Flow Formulation')
   ;
     Selection = 3,
     write(' Lower Bound Formulation')
   ;
```

```
      write(' Upper Bound Formulation')
    ).
/*-----------------------------------------------------------------*/

objective(Selection) :-              % Get objective function
    nl,nl,
    write(' ..Objective Maximize'),
    nl,nl,
    ctr_set(1,1),  % Counter one generates path number.
    ctr_set(3,1),  % Initialize counter three to 1
                   % to keep track of how many terms
                   % are printed in a line in the objective
                   % function.
    find_objective(s,t,Selection).


find_objective(Start,Goal,Selection) :-
    depth_first([Start],Goal,Path),
    find_reliability(Path,Rel),
    ctr_inc(1,Pnbr),                 % Get current path number and
                                     % increment the counter one by one.
    print_objective(Pnbr,Rel,Selection),
    make_arc_path_matrix(Path,Pnbr),  % Make arc-path incidence
                                      % matrix.
    fail.
find_objective(_,_,_).


print_objective(Pnbr,Rel,Selection) :-
    tab(1),
    ( Pnbr = 1
    ;
      write('+ ')
    ),
    ( ctr_inc(3,Value),     % Get current variable number and
                            % increment the counter three by 1.
      Value > 4,
      Mod_Value is Value mod 4,   % If the remainder of Value
      Mod_Value = 1,              % divided by 4 is 1, then skip
      nl,                         % to next line.
      write(' ')
    ;
      true
    ),
    ( Selection = 3,              % If finding lower bound (Sel = 3),
      write(Rel)                  % print reliability.
    ;
      true                        % Otherwise, do nothing.
    ),
    write(' f'),                  % Print path flow variable.
    write(Pnbr), !.
```

```
/*-----------------------------------------------------------------*/

 make_arc_path_matrix([],_).
 make_arc_path_matrix(Arc_List,Pnbr) :-   % Seperate the elements,
                                          % arcs, in the path and
                                          % store each arc with
                                          % associated path number
                                          % to form arc-path incidence
    get_arc(Arc_List,Arc,Rem_List),       % matrix.
    cap(Arc,Capacity),
    ( number(Capacity),
      recordh(matrix,Arc,arc_path_matrix(Arc,Pnbr))
    ;
      true
    ),
    make_arc_path_matrix(Rem_List,Pnbr), !.


 get_arc([Head|[H|Rest]],Head,Rem_List) :-
    Head \= 's',                        % Ignore 's' and 't'
    Head \= 't',
    ( Rest = [],
      Rem_List = []
    ;
      Rem_List = [H|Rest]
    ), !.

 get_arc([Head|Rest],Arc,Rem_List) :-
    get_arc(Rest,Arc,Rem_List).         % Get one arc at a time
                                        % that is in the path.


/*-----------------------------------------------------------------*/

 constraints(Selection) :-              % Get constraints function.
     nl, nl,
     write(' ..Constraints'),
     asserta(init_string($$)),          % Initialize to empty string.
     find_all_arc_lists(Arc_List),
     sort(Arc_List,Sor),                % Sort Arc_List in ascending order.
     make_arc_array(Sor),               % Make sorted arc_list into
                                        % arc array.
     generate_constraints(Selection).

 find_all_arc_lists(_) :-               % Find all arcs that are in the
                                        % arc-path incidence matrix.
     retrieveh(matrix,_,arc_path_matrix(AN,_)),
     int_text(AN, Arc_String),
     [! concat([$0000$, Arc_String, $,$], New_Arc_String),
        retract(init_string(Init)),
```

```
      ( string_search(New_Arc_String,Init,_), % Do not include the
        asserta(init_string(Init))            % duplicate arc string.
      ;
        concat(New_Arc_String, Init, New_String), % Append the new
        asserta(init_string(New_String))          % string.
      )
    !],
    fail.


find_all_arc_lists(Final) :-
    retract(init_string(Main_String)),
    string_length(Main_String, Length),
    dec(Length,Pos),
    substring(Main_String,0,Pos,New_String),
    concat([$[$, New_String, $]$], Output_String),
    string_term(Output_String, Final), !.     % Change string into
                                              % a list.


/*-------------------------------------------------------------------*/

make_arc_array([]).                 % Seperate the arc_list and
make_arc_array([Anbr|Rest]) :-      % put it into an array format.
    assertz(arc_array(Anbr)),
    make_arc_array(Rest).


/*-------------------------------------------------------------------*/

generate_constraints(Selection) :-
    retract(arc_array(Anbr)),
    [! cap(Anbr,Capacity),
       generate_constraint_inequality(Anbr),
       write(' '),
       write('<= '),
       write(' '),
       ( Selection = 4,
         prob(Anbr,Pb),
         Expected_Cap is Capacity * Pb,
         write(Expected_Cap)
       ;
         write(Capacity)
       )
    !],
    fail.

generate_constraints(_).

generate_constraint_inequality(Anbr) :-
    nl, nl,
```

```
    write(' Arc '),
    write(Anbr),
    write(': '),
    ctr_set(10,0),       % First time flag; this is used to
                             % control when to print '+' in
                             % the constraint equation.
    ctr_set(4,1),       % Initialize counter four to 1;
                             % this counter keeps track of
                             % how many terms are printed in
                             % the constraint equation.
    output_paths_containing_Anbr(Anbr).


output_paths_containing_Anbr(Anbr) :-
    removeh(matrix,Anbr,arc_path_matrix(Anbr,Pnbr)),
    [!  ( ctr_inc(10,Flag),
          Flag = 0              % If first term, don't print '+'.
        ;
          write(' + ')
        ),
        ( ctr_inc(4,Value),     % Get current term number and
                                % increment the counter four by 1
          Value > 7,
          Mod_Value is Value mod 7,
          Mod_Value = 1,
          nl, write('         ')
        ;
          true
        ),
        write('f'),
        write(Pnbr)
    !],
    fail.

output_paths_containing_Anbr(_).

/*****************************************************************/
/* 'investment_formulations' generates formulations of invest- */
/* ment strategy model 1 and 2 to improve the lower bound.     */
/*****************************************************************/


investment_formulations(Selection,Where) :-
    ( Where = 1,
      get_investment_model(Selection)
    ;
      executing_message,
      ( Selection = 5,
        stdout('output5.lp',get_investment_model(Selection))
      ;
        stdout('output6.lp',get_investment_model(Selection))
```

```
      ),
      exit_popup,
      done_message(Selection)
    ).

get_investment_model(Selection) :-
    invest_model_title(Selection),
    invest_model_objective(Selection),
    invest_model_constraints(Selection),
    nl, nl,
    write('* ------ end ------ *'),
    nl,nl,nl,nl.

invest_model_title(Selection) :-
   write(' ..Title'),
   nl,nl,
   ( Selection = 5,
     write(' Investment Strategy Model 1')
   ;
     write(' Investment Strategy Model 2')
   ).

invest_model_objective(Selection) :-
    nl, nl,
    write(' ..Objective Maximize'),
    nl, nl,
    ctr_set(1,1),                       % path number
    ctr_set(3,1),                       % no. of terms in a line
    find_invest_model_objective(s,t,Selection).

find_invest_model_objective(Start,Goal,_) :-
    depth_first([Start],Goal,Path),
    find_reliability(Path,Rel),
    ctr_inc(1,Pnbr),
    output_invest_model_objective_variables(Pnbr,Rel),
    make_arc_path_matrix(Path,Pnbr),
    fail.

find_invest_model_objective(_,_,Selection) :-
    asserta(init_invest_string($$)),
    get_investment_variables(Invest_Vars),
    sort(Invest_Vars,Sorted_Vars),
    make_cap_array(Sorted_Vars),
    nl,
    ctr_set(2,1),
    ctr_set(10,0),                          % First time flag.
    ( Selection = 5
    ;
      write(' [ ')
    ),
    output_investment_variables(Selection),
```

```
    ( Selection = 5
    ;
      write(' ] ')
    ).

find_invest_model_objective(_,_,_).

make_cap_array([]).
make_cap_array([Anbr|Rest]) :-
    assertz(cap_array(Anbr)),
    make_cap_array(Rest).


output_invest_model_objective_variables(Pnbr,Rel) :-
    write(' '),
    write(Rel),
    write(' f'),
    write(Pnbr),
    write(' +'),
    ( ctr_inc(3,Value),
      Mod_Value is Value mod 4,
      Mod_Value = 0,
      nl
    ;
      true
    ), !.


get_investment_variables(_) :-
    cap(AN,Capacity),
    [! ( not number(Capacity)
       ;
         int_text(AN, Arc_String),
         concat([$0000$, Arc_String, $,$], New_Arc_String),
         retract(init_invest_string(Init)),
         concat(New_Arc_String, Init, New_String),
         asserta(init_invest_string(New_String))
)
    !],
    fail.


get_investment_variables(Final) :-
    retract(init_invest_string(Main_String)),
    string_length(Main_String, Length),
    dec(Length,Pos),
    substring(Main_String,0,Pos,New_String),
    concat([$[$, New_String, $]$], Output_String),
    string_term(Output_String, Final), !.
```

```
output_investment_variables(Selection) :-

   cap_array(Anbr),
   [!  ( ctr_inc(10,Flag),
         Flag = 0
       ;
         write(' +')
       ),
       ( ctr_inc(2,Value),
         Value > 7,
         Mod_Value is Value mod 7,
         Mod_Value = 1,
         nl
       ;
         true
       ),
       ( Selection = 5,
         write(' 0 d')
       ;
         write(' 0 g')
       )
   !],
   write(Anbr),
   fail.

output_investment_variables(_).

/*-------------------------------------------------------*/


invest_model_constraints(Selection) :-

    nl, nl,
    write(' ..Constraints'),
    asserta(init_string($$)),
    find_all_arc_lists(Arc_List),
    sort(Arc_List,Sor),
    make_arc_array(Sor),
    generate_arc_constraints(Selection),
    ctr_set(10,0),              % counter for budget term
    nl, nl,
    write(' Budget: '),
    generate_budget_constraint(Selection).


generate_arc_constraints(Selection) :-
    retract(arc_array(Anbr)),
    [! cap(Anbr,Capacity),
       generate_constraint_inequality(Anbr),
       ( Selection = 5,
         write(' - d')
```

```
        ;
          write(' - '),
          invest(Anbr,Amount),
          write(Amount),
          write(' g')
        ),
        write(Anbr),
        write(' '),
        write('<= '),
        write(' '),
        write(Capacity)
    !],
    fail.

generate_arc_constraints(_).


generate_budget_constraint(Selection) :-
   retract(cap_array(Anbr)),
   [! cost(Anbr,Unit_cost),
      ( Selection = 5,
        Cost is Unit_cost
      ;
        invest(Anbr,Amount),
        Cost is Unit_cost * Amount
      ),
      print_budget_terms(Anbr,Cost,Selection)
   !],
    fail.

generate_budget_constraint(_) :-
    ctr_set(5,1),
    budget(Budget),
    write(' '),
    write(' <= '),
    write(Budget).


print_budget_terms(Anbr,Cost,Selection) :-
   ( ctr_inc(10,Flag),
     Flag = 0
   ;
     write(' + ')
   ),
   ( ctr_inc(5,Value),
     Value > 5,
     Mod_Value is Value mod 5,
     Mod_Value = 1,
     nl, write('         ')
   ;
     true
```

```
    ),
    write(Cost),
    ( Selection = 5,
      write(' d')
    ;
      write(' g')
    ),
    write(Anbr).

/*-------------------------------------------------------------*/
 :- reconsult('windows.ari').
/* ---------------------- end ------------------------------- */
```

A.2 WINDOWS.ARI: Dialog Windows

```
/*********************************************************************/
/*                                                                   */
/*                      WINDOWS.ARI                                  */
/*                                                                   */
/*  This file contains windows to communicate with the user.         */
/*                                                                   */
/*********************************************************************/

/*-------------------------------------------------------------------*/
/*  Introduction Screen                                              */
/*-------------------------------------------------------------------*/

windows :-
  cls,
  define_window(program_title,'',(23,0),(23,79),(91,0)),
  define_window(intro,'',(0,0),(22,79),(26,0)),
  current_window(_,program_title),
  tmove(0,12),
  write('  FORMULA   Version 1.0       AFIT   October, 1988'),
  current_window(_,intro),
  define_intro_window.

 define_intro_window :-
  nl,nl,
  tab(25),
  write('************************'),
  nl,tab(25),
  write('*     F O R M U L A     *'),
  nl,tab(25),
  write('************************'),
  nl,nl,nl,
  tab(11),
  write('This program finds all paths in the network from '),
  nl,tab(11),
  write('source to sink and calculates all paths reliabilities.'),
  nl,tab(11),
  write('Then, it generates five mathematical programming '),
  nl,tab(11),
  write('models that will assist in analyzing the performance '),
  nl,tab(11),
  write('of the network and in determining the investment '),
  nl,tab(11),
  write('strategy to improve the performance of the network.'),
  nl,tab(11),
  write('These models are developed based on the arc-path '),
  nl,tab(11),
  write('incidence matrix built from the description of the '),
  nl,tab(11),
  write('network in the input file. '),
```

```
  nl,nl,tab(11),
  write('PLEASE make sure the input file contains correct'),
  nl,tab(11),
  write('description of the network to be analyzed.'),
  nl,nl,tab(23),
  write('Press any key to continue. '),
  get0(_),
  cls.


/*-------------------------------------------------------------*/
/* Asks for the input file name. If the file name is not found, */
/* the program prints the error message; otherwise, consults    */
/* the input file.                                              */
/*-------------------------------------------------------------*/

open_input_datafile :-
  create_popup(query1,(7,20),(14,60),(62,-62)),
  write(' Please type in your input file name.   '),
  tmove(3,2),
  write(' > '),
  read_line(0,File),
  (
    consult_file(File),
    exit_popup
  ;
    display_filename_error,
    exit_popup,
    open_input_datafile
  ).

consult_file(File) :-
   stdin(File,_),
   consult(File).

display_filename_error :-
  create_popup(error1,(16,20),(21,60),(79,-79)),
  write('     Error: File not found. '),
  put(7),
  nl, nl,
  write(' Type in any key to continue or '),
  nl,
  write(' press RETURN to exit. '),
  get0(Reply),
  ( Reply = 13,
    exit_popup,
    exit_popup,
    clear_windows
  ;
    exit_popup
  ).
```

```
/*-----------------------------------------------------------------*/
/* Ask user what to do.            */
/*-----------------------------------------------------------------*/

get_selection_number(Selection) :-
  create_popup(query2,(3,12),(19,68),(62,-62)),
  tmove(1,16),
  write('How may I help you?'),
  tmove(4,2),
  write('1. Find all paths and calculate path reliabilities.'),
  tmove(5,2),
  write('2. Generate the Maximum Flow Formulation.'),
  tmove(6,2),
  write('3. Generate the Lower Bound Formulation. '),
  tmove(7,2),
  write('4. Generate the Upper Bound Formulation. '),
  tmove(8,2),
  write('5. Generate the Investment Strategy Model 1. '),
  tmove(9,2),
  write('6. Generate the Investment Strategy Model 2. '),
  tmove(10,2),
  write('7. Exit'),
  tmove(13,18),
  write('Type in number >  '),
  get0(Choice),           % The selection chosen is in ASCII code,
  exit_popup,             % that is one is represented as 49, two is
  Sel_Nbr is Choice - 48, % represented as 50, etc. Thus, 48 is
  ( Sel_Nbr >= 1,         % substracted to make it back to regular
    Sel_Nbr =< 7,         % arabic number.
    Selection = Sel_Nbr
  ;
    put(7),
    get_selection_number(Selection)
  ).


/*-----------------------------------------------------------------*/
/* Asks user where to display the output.                          */
/*-----------------------------------------------------------------*/

get_where_to_send_output(Where) :-
  create_popup(query3,(5,20),(16,60),(62,-62)),
  tmove(1,8),
  write('Where do you want the'),
  tmove(2,8),
  write('output displayed ?'),
  tmove(4,11),
  write('1. Screen'),
  tmove(6,11),
```

```
  write('2. File'),
  tmove(8,8),
  write('Type in Number >  '),
  get0(Choice),
  exit_popup,
  Sel_Nbr is Choice - 48,
  ( Sel_Nbr >= 1,
    Sel_Nbr =< 2,
    Where = Sel_Nbr
  ;
    put(7),
    get_where_to_send_output(Where)
  ).


/*-----------------------------------------------------------------*/
/* Asks user to run the program again with same input file.        */
/*-----------------------------------------------------------------*/

get_run_again_reply(Reply):-
  create_popup(query4,(20,0),(22,79),(62,-62)),
  write(' Do you want to run the program again (y or n) ?  '),
  get0(User_Reply),
  exit_popup,
  (
    ( User_Reply = 121;      % y
      User_Reply = 110      % n
    ),
    Reply = User_Reply
  ;
    put(7),
    get_run_again_reply(Reply)
  ).


/*-----------------------------------------------------------------*/
/* Prints 'executing' message while output is sent to an output    */
/* file.                                                           */
/*-----------------------------------------------------------------*/

executing_message :-
  create_popup('',(11,25),(14,50),(207,79)),
  write('      Executing ... '),
  nl,
  write('      Please Wait.').


/*-----------------------------------------------------------------*/
/* Prints 'Done' message after output is sent to an output file.   */
/*-----------------------------------------------------------------*/
```

```
done_message(Output_File) :-
  create_popup('',(11,20),(14,57),(58,58)),
  write('              Done. '),
  nl,
  write('  Output was sent to "output'),
  write(Output_File),
  write('.lp".').


/*-----------------------------------------------------------------*/
/* Clears all windows before exiting(quiting) the program.         */
/*-----------------------------------------------------------------*/

clear_windows :-
  delete_window(program_title),
  delete_window(intro),
  abolish(arc/2),
  abolish(prob/2),
  abolish(cap/2),
  abolish(cost/2),
  abolish(budget/1),
  removeallh(matrix),
  current_window(_,main).

/*------------------------------ end -------------------------------*/
```

Appendix B. *FORMULA User's Manual*

This manual explains how to use Prolog program, FORMULA Version 1.0. This program consists of two files:

* FORMULA.ARI - Contains the main computer program.

* WINDOWS.ARI - Contains window dialog boxes.

B.1 Required Equipment

Currently, FORMULA requires ARITY/PROLOG interpreter program to run. The interpreter and FORMULA can be installed on an IBM-XT/AT compatible microcomputer. The computer with at least 512 kilobytes of random access memory and 20 megabyte hard disk is desired.

B.2 Running the FORMULA

Assumming both ARITY/PROLOG interpreter and FORMULA are installed on your computer, start the interpreter by typing "**API**". When "?-" prompt appears, consult your program by typing "**consult('formula.ari').**" After the program has been consulted correctly, type in "**go.**" to start the program FORMULA. The program proceeds through the following steps:

1) First, it displays an introductory screen briefly describing what this program can do. (Just hit any key to go on.)

2) Next, it asks for the name of an input data file which contains the description of the network to be analyzed. (Type in the exact name of the input file and hit return.)

3) After an input file name has been typed in, the program displays a menu window from which you can choose to generate a specific output (Select number 1, 2, 3, 4, 5, 6, or 7):

1. Find all paths and calculate path reliabilities.

2. Generate the Maximum Flow Formulation.

3. Generate the Lower Bound Formulation.

4. Generate the Upper Bound Formulation.

5. Generate the Investment Strategy Model 1.

6. Generate the Investment Strategy Model 2.

7. Exit.

4) It then asks where to send the output. If you just want to screen the output, choose 1; otherwise, choose 2 to save the output in a file. The output filename is automatically generated by the program. When the user requests the output to be sent to a file, the output of paths and reliabilities is sent to 'output1.lp', maximum flow to 'output2.lp', lower bound to 'output3.lp', upper bound to 'output4.lp', investment strategy model 1 to 'output5.lp', and investment strategy model 2 to 'output6.lp'. (Select 1 or 2.)

5) After the output has been generated, the program asks if you want to run the program again. If you are interested in getting other output, type in "y"; otherwise type in "n" which exits the program. (Type in y or n.)

B.3 Input

In preparing the input data, you have to follow the following procedures:

1) If the network contains stochastic and/or capacitated nodes, convert the nodes to a dummy arc joined by two nodes. The dummy arc represents the stochastic and/or capacitated node.

2) Introduce an artificial single source and single sink. The source and sink must be named s and t, respectively. Connect all source nodes in the network to s, and all sink nodes to t.

3) Draw a revised network, and assign arc numbers. It has been found very helpful if you number the dummy arcs representing the node with the same node number. Then number the remaining arcs starting with one number higher than the number of nodes in the network. Thus, for example, if you have 20 nodes in the original network, number the remaining arcs starting with 21.

4) Using a text editor (Arity/Prolog comes with its own editor), prepare the input data by typing in the description of revised network that is to be analyzed. The input consists of six data sets: description of an arc relationship with respect to one another, the survival probability of each arc, the capacity of each arc, the cost of improving each arc by one unit of capacity, the predetermined amount of capacity increase in each component, and total budget available for investment. These input are described as *facts* in Prolog as follows:

Input	Description
`arc(Arc1,Arc2).`	`Arc1 is the parent of Arc2 (Arc1 precedes Arc2).`
`prob(A,Pb).`	`The survival probability of arc A is Pb.`
`cap(A,Cp).`	`The capacity of arc A is Cp. The unlimited (infinite) capacity is denoted by '*'.`
`cost(A,Cs).`	`The cost of increasing one unit of capacity in arc A is Cs.`
`invest(A,Am).`	`The predetermined amount of capacity increase for arc A is Am.`
`budget(B).`	`Total budget available for investment is B.`

When defining arc relationships, you must define the relationship of s and t

with respect to other arcs. So "`arc(s,2).`" defines that the arc 2 is incidence to node `s`; that is, the arc 2 leaves the node `s`. In a likely manner, "`arc(22,t).`" defines that the arc 22 arrives at node `t`. In addition to specifying the probability of each arc, the probability of `s` and `t` must be defined as 1. Thus, in the probability description, make sure you include "`prob(s,1)`" and "`prob(t,1)`".

5) Depending on your need, some of the input data may be omitted. Refer to the table below to see exactly which data sets are required to get the desired output. For example, if you are only interested in finding paths and their reliabilities, all you need is arc relationships and survival probabilities.

For this output:	You must have the following data:
Paths and Reliabilities	arc(X,Y). prob(X,Y).
Maximum Flow, Lower Bound, or Upper Bound	arc(X,Y). prob(X,Y). cap(X,Y).
Improvement Strategy 1	arc(X,Y). prob(X,Y). cap(X,Y). cost(X,Y). budget(X,Y).
Improvement Strategy 2	arc(X,Y). prob(X,Y). cap(X,Y). cost(X,Y). budget(X,Y). invest(X,Y).

6) When using Arity/Prolog, it is customary to name the file with '.ari' extension. So name your input file with '.ari' extension.

B.4 Example

To illustrate how to prepare the input data, consider a network shown in Figure B.1. This network contains 4 nodes and 3 arcs. It has multiple sources, 1 and 2, and a single sink, 4.

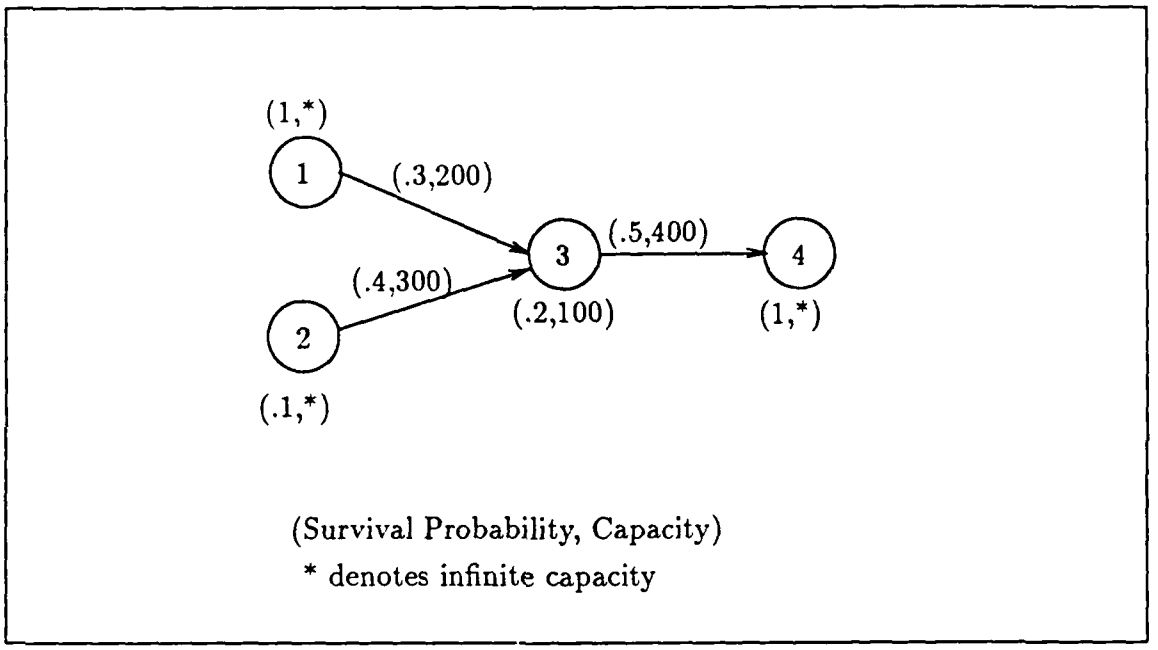


Figure B.1. Sample Network

The cost of increasing one unit of capacity is as follows: node 3 = 10, $arc_{1,3}$ = 20, $arc_{2,3}$ = 30, and $arc_{3,4}$ = 40. The predetermined amount of capacity increase is 5 *units for all components*, and the total budget available for investment is $1000.

The revised network is shown in Figure B.2.

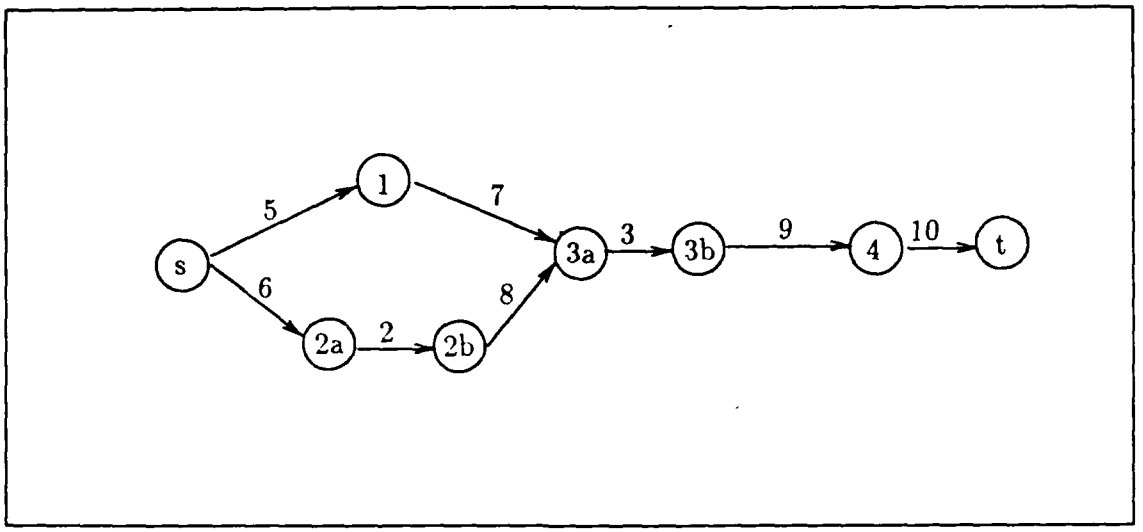


Figure B.2. Revised Network Shown in Figure B.1

Since node 1 and 4 are not stochastic and not capacitated, they do not need to be represented as arcs. A dummy arc representing a stochastic or capacitated node is assigned the same number as its node number. When numbering the arcs, the number 1 and 4 are not used, since 1 and 4 are not represented as arcs. The remaining arcs are numbered starting with 5. Now, refering to the revised network shown in Figure B.2, the input data can be prepared as follows:

```
% Arc Relationship
   arc(s,5).
   arc(s,6).
   arc(2,8).
   arc(3,9).
   arc(5,7).
   arc(6,2).
   arc(7,3).
   arc(8,3).
   arc(9,10).
   arc(10,t).
```

```
% Survival Probabilities
   prob(s,1).
   prob(2,0.1).
   prob(3,0.2).
   prob(5,1).
   prob(6,1).
   prob(7,0.3).
   prob(8,0.4).
   prob(9,0.5).
   prob(10,1).
   prob(t,1).

% Capacity
   cap(3,100).
   cap(5,*).
   cap(6,*).
   cap(7,200).
   cap(8,300).
   cap(9,400).
   cap(10,*).

% Cost of Increasing One unit of Capacity
   cost(3,10).
   cost(7,20).
   cost(8,30).
   cost(9,40).

% Predetermined Amount of Capacity Increase
   invest(_,5).

% Budget Available
   budget(1000).

% --- end--- %
```

Any line that starts with a % sign is a comment line, and it is ignored by the interpreter. The underscore (_) in "`invest(_,5).`" denotes *all components*. Thus, "`invest(_,5).`" denotes the predetermined amount of capacity increase for all components is 5 uints.

B.5 Output

An example of output1.lp, containing paths and path reliabilities, are shown below:

```
*****************************************************
* Following is a list of all paths from "s" to "t" *
* of the network described in the input data file. *
*****************************************************

  Path1: s 5 7 3 9 10 t
  Reliability:  0.003

  Path2: s 6 2 8 3 9 10 t
  Reliability:  0.004

* ----- end ----- *
```

The numbers in the paths are the arc numbers from revised network shown in Figure B.2.

The rest of the outputs, containing mathematical programming models, are in the same format as the input format of LP/MIP-83. Each output consists of ..Title, ..Objective Maximize, and ..Constraints section. An example of output2.lp, containing the maximum flow formulation, is shown below:

```
..Title

Maximum Flow Formulation 2

..Objective Maximize

f1 + f2

..Constraints

Arc 3: f1 + f2 <= 100

Arc 7: f1 <= 200
```

```
Arc 8: f2 <= 300

Arc 9: f1 + f2 <= 400

* ----- end ----- *
```

B.6 LP/MIP-83 Commands

All models, except the Investment Strategy Model 2, can be solved using either LP83 or MIP83. The Investment Strategy Model 2 can only be solved by MIP83, because it requires integer solutions. The following are some of the commands to run the LP/MIP-83:

a) *c* :> lp83 a:output2

Find all solutions to the linear programming model stored in 'output2.lp' in 'a' drive, and display the solutions on the screen.

b) c :> lp83 a:output2 output a:list

Same as a) above, except send the solutions to the output file named 'list.prn', in 'a' drive. The extension, '.prn' is automatically added.

c) *c* :> lp83 a:output2 output a:list alternate 1

Same as b) above, except find only one solution.

d) *c* :> mip83 a:output6

Find all solutions to the mixed integer programming model stored in 'output6.lp' in 'a' drive, and display the solutions on the screen.

B.7 Helpful Comments

1) The program, FORMULA, has been tested and successfully generated formulations for the network containing 70 nodes, 112 arcs, and 2198 paths. Since

the capacity of LP/MIP-83 is approximately 1200 variables (paths), any problem bigger than the capacity of LP/MIP-83, must be solved using other mathematical programming packages, such as MINOS or SAS. Of course preparing an input file for these packages will be different from that of LP/MIP-83.

2) If you want to stop the execution (or if the computer is hung up), press 'control' and 'break' simultaneously. When '?-' appears, type 'clear_windows.' and/or 'exit_popup.'. It will get you to the main window of Arity/Prolog. Before starting the program again, you must erase the datafile by typing 'Restore.' Otherwise, you will get errorneous output.

3) Any questions about the Arity/Prolog interpreter or general questions about the Arity/Prolog, refer to References [1] and [2]. Any questions on LP/MIP-83, refer to reference [3].

B.8 References

1 Arity Corporation. *The Arity/Prolog Language Reference Manual.* Concord, Masschusetts, 1988.

2 Arity Corporation. *Using the Arity/Prolog Compiler and Interpreter.* Concord, Masschusetts, 1987.

3 Sunset Software. *LP/MIP 83: A Professional Linear and Mixed Integer Programming System.* San Marino, California, 1985.

Appendix C. *Experimental Results for Example Network*

C.1 Input File to FORMULA

```
/*============================================================*/
/*                  INPUT DATA FILE: N1.ARI                   */
/*============================================================*/
/*-- arc relationship --*/
 arc(s,1).
 arc(s,2).
 arc(1,3).
 arc(1,4).
 arc(2,5).
 arc(3,6).
 arc(4,7).
 arc(5,7).
 arc(7,t).
 arc(6,t).
/*-- arc survival probability --*/
 prob(s,1).
 prob(1,1).
 prob(2,0.1).
 prob(3,1).
 prob(4,1).
 prob(5,0.5).
 prob(6,0.9).
 prob(7,1).
 prob(t,1).
/*-- arc capacity --*/
 cap(1,5).
 cap(2,6).
 cap(3,2).
 cap(4,4).
 cap(5,5).
 cap(6,4).
 cap(7,7).
/*-- investment cost --*/
 cost(1,50).
 cost(2,60).
 cost(3,20).
 cost(4,40).
 cost(5,50).
 cost(6,40).
 cost(7,70).
/*-- predetermined amount of investment --*/
 invest(1,5).
 invest(2,5).
 invest(3,5).
```

```
 invest(4,5).
 invest(5,5).
 invest(6,5).
 invest(7,5).
/*-- budget --*/
 budget(1000).
/*--- end of data ---*/
```

C.2 A Listing of Paths and Path Reliabilities

```
*********************************************************
*  Following is a list of all paths from "s" to "t"     *
*  of the network described in the input data file.      *
*********************************************************


  Path 1: s 1 3 6 t
  Reliability: 0.9

  Path 2: s 1 4 7 t
  Reliability: 1

  Path 3: s 2 5 7 t
  Reliability: 0.05

* ------ end ------ *
```

C.3 Solution to Maximum Flow Model

```
LP83 a:n1max2 output a:n1max2out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Maximum Flow Formulation

 ..Objective Maximize

  f1 +  f2 +  f3

 ..Constraints

 Arc 1: f1 + f2 <=  5

 Arc 2: f3 <=  6

 Arc 3: f1 <=  2

 Arc 4: f2 <=  4

 Arc 5: f3 <=  5

 Arc 6: f1 <=  4

 Arc 7: f2 + f3 <=  7

* ------ end ------ *


Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  301K bytes.
  Objective Function is MAXIMIZED.
  Variables:          3
  Constraints:        7
     7 LE,   0 EQ,   0 GE.
  Non-zero LP elements:        9
  Disk Space:      0K bytes.
  Page Space:      0K bytes.
  Capacity:  2.7% used.
  Estimated Time: 00:00:00
```

Iter 3
Solution Time: 00:00:00
A L T E R N A T E S O L U T I O N S

File: n1max2 11/19/88 00:36:02 Page 1-1
SOLUTION (Maximized): 9.0000 Maximum Flow Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f1	2.0000	1.0000	f2	3.0000	1.0000
f3	4.0000	1.0000			

File: n1max2 11/19/88 00:36:02 Page 1-2
CONSTRAINTS: Maximum Flow Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 1	5.0000 <	5.0000	Arc 2	4.0000 <	6.0000
Arc 3	2.0000 <	2.0000	Arc 4	3.0000 <	4.0000
Arc 5	4.0000 <	5.0000	Arc 6	2.0000 <	4.0000
Arc 7	7.0000 <	7.0000			

Total Error: 0.000000

C.4 Solution to Lower Bound Model

```
LP83 a:n1low output a:n1lowsen alternate 1 costanalysis yes marginanalysis yes

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Lower Bound Formulation

 ..Objective Maximize

 0.9 f1 + 1 f2 + 0.05 f3

 ..Constraints

 Arc 1: f1 + f2 <=  5

 Arc 2: f3 <=  6

 Arc 3: f1 <=  2

 Arc 4: f2 <=  4

 Arc 5: f3 <=  5

 Arc 6: f1 <=  4

 Arc 7: f2 + f3 <=  7

* ------ end ------ *


Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  301K bytes.
  Objective Function is MAXIMIZED.
  Variables:          3
  Constraints:        7
     7 LE,    0 EQ,    0 GE.
  Non-zero LP elements:        9
  Disk Space:      0K bytes.
  Page Space:      0K bytes.
  Capacity:  2.7% used.
  Estimated Time: 00:00:00
```

Iter 3
Solution Time: 00:00:01
U N I Q U E S O L U T I O N

File: nilow 12/02/88 04:25:28 Page 1-1
SOLUTION (Maximized): 5.0500 Lower Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
I f1	1.0000	0.9000	I f2	4.0000	1.0000
I f3	3.0000	0.0500			

File: nilow 12/02/88 04:25:28 Page 1-2
CONSTRAINTS: Lower Bound Formulation

Constraint	Activity		RHS	Constraint	Activity		RHS
Arc 1	5.0000	<	5.0000	I Arc 2	3.0000	<	6.0000
I Arc 3	1.0000	<	2.0000	Arc 4	4.0000	<	4.0000
I Arc 5	3.0000	<	5.0000	I Arc 6	1.0000	<	4.0000
Arc 7	7.0000	<	7.0000				

Total Error: 0.000000

File: nilow 12/02/88 04:25:28 Page 1-3
COST ANALYSIS: Lower Bound Formulation

Variable	Stable Cost Range	Variable to Change	Variable	Stable Cost Range	Variable to Change
Upper	0.9500	Arc 4	Upper	UNBOUNDED	
f1	0.9000		f2	1.0000	
Lower	0.0000	Arc 1	Lower	0.9500	Arc 4
Reduced Cost		0.0000	Reduced Cost		0.0000
Upper	0.1000	Arc 4			
f3	0.0500				
Lower	0.0000	Arc 7			
Reduced Cost		0.0000			

File: n1low 12/02/88 04:25:28 Page 1-4
MARGINAL ANALYSIS: Lower Bound Formulation

Constraint at limit	Value	Constraint at limit	Value
Arc 1 <	5.0000	Arc 4 <	4.0000
Increases objective ... by	0.9000	Increases objective ... by	0.0500
Upper Limit.		Upper Limit.	
New limit ..	6.0000	New limit ..	5.0000
New optimum ...	5.9500	New optimum ...	5.1000
Forced to limit	Arc 3	Forced to limit	f1
Lower Limit.		Lower Limit.	
New limit ..	4.0000	New limit ..	3.0000
New optimum ...	4.1500	New optimum ...	5.0000
Forced to limit	f1	Forced to limit	Arc 3

File: n1low 12/02/88 04:25:28 Page 1-5
MARGINAL ANALYSIS: Lower Bound Formulation

Constraint at limit	Value
Arc 7 <	7.0000
Increases objective ... by	0.0500
Upper Limit.	
New limit ..	9.0000
New optimum ...	5.1500
Forced to limit	Arc 5
Lower Limit.	
New limit ..	4.0000
New optimum ...	4.9000
Forced to limit	f3

C.5 Solution to Upper Bound Model

```
LP83 a:n1up2 output a:n1up2out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Upper Bound Formulation

 ..Objective Maximize

  f1 +  f2 +  f3

 ..Constraints

 Arc 1: f1 + f2 <=  5

 Arc 2: f3 <=  0.6

 Arc 3: f1 <=  2

 Arc 4: f2 <=  4

 Arc 5: f3 <=  2.5

 Arc 6: f1 <=  3.6

 Arc 7: f2 + f3 <=  7

* ------ end ------ *


Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  301K bytes.
  Objective Function is MAXIMIZED.
  Variables:          3
  Constraints:        7
     7 LE,    0 EQ,    0 GE.
  Non-zero LP elements:        9
  Disk Space:      0K bytes.
  Page Space:      0K bytes.
  Capacity:  2.7% used.
  Estimated Time: 00:00:00
```

Iter 3
Solution Time: 00:00:00
A L T E R N A T E S O L U T I O N S

File: n1up2 11/19/88 00:36:50 Page 1-1
SOLUTION (Maximized): 5.6000 Upper Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f1	2.0000	1.0000	f2	3.0000	1.0000
f3	0.6000	1.0000			

File: n1up2 11/19/88 00:36:50 Page 1-2
CONSTRAINTS: Upper Bound Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 1	5.0000 <	5.0000	Arc 2	0.6000 <	0.6000
Arc 3	2.0000 <	2.0000	Arc 4	3.0000 <	4.0000
Arc 5	0.6000 <	2.5000	Arc 6	2.0000 <	3.6000
Arc 7	3.6000 <	7.0000			

Total Error: 0.000000

C.6 Solution to Investment Strategy Model 1

```
LP83 a:n1imp1 output a:n1imp1out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Investment Strategy Model 1

 ..Objective Maximize

 0.9 f1 + 1 f2 + 0.05 f3 +
 0 d1 + 0 d2 + 0 d3 + 0 d4 + 0 d5 + 0 d6 + 0 d7

 ..Constraints

 Arc 1: f1 + f2 - d1 <=  5

 Arc 2: f3 - d2 <=  6

 Arc 3: f1 - d3 <=  2

 Arc 4: f2 - d4 <=  4

 Arc 5: f3 - d5 <=  5

 Arc 6: f1 - d6 <=  4

 Arc 7: f2 + f3 - d7 <=  7

 Budget: 50 d1 + 60 d2 + 20 d3 + 40 d4 + 50 d5 +
         40 d6 + 70 d7  <= 1000

* ------ end ------ *



 Statistics-
   LP83 Version 5.00a
   Machine memory:  640K bytes.
   Pagable memory:  301K bytes.
   Objective Function is MAXIMIZED.
   Variables:        10
   Constraints:       8
```

```
    8 LE,   0 EQ,   0 GE.
  Non-zero LP elements:      23
  Disk Space:      0K bytes.
  Page Space:      1K bytes.
  Capacity:  3.3% used.
  Estimated Time: 00:00:00

Iter    7
Solution Time:  00:00:00
U N I Q U E   S O L U T I O N
```

File: n1imp1 11/19/88 00:24:11 Page 1-1

SOLUTION (Maximized): 15.0182 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
I f1	8.9091	0.9000	I f2	7.0000	1.0000
I f3	0.0000	0.0500	I d1	10.9091	0.0000
I d2	0.0000	0.0000	I d3	6.9091	0.0000
I d4	3.0000	0.0000	I d5	0.0000	0.0000
I d6	4.9091	0.0000	I d7	0.0000	0.0000

File: n1imp1 11/19/88 00:24:11 Page 1-2

CONSTRAINTS: Investment Strategy Model 1

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 1	5.0000 <	5.0000	I Arc 2	0.0000 <	6.0000
Arc 3	2.0000 <	2.0000	Arc 4	4.0000 <	4.0000
I Arc 5	0.0000 <	5.0000	Arc 6	4.0000 <	4.0000
Arc 7	7.0000 <	7.0000	Budget	1,000.0000 <	1,000.0000

Total Error: 0.000000

C.7 Solution to Investment Strategy Model 2

```
MIP83 a:n1imp2 output a:n1imp2out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Investment Strategy Model 2

 ..Objective Maximize

 0.9 f1 + 1 f2 + 0.05 f3 +
 [  0 g1 + 0 g2 + 0 g3 + 0 g4 + 0 g5 + 0 g6 + 0 g7 ]

 ..Constraints

 Arc 1: f1 + f2 - 5 g1 <=  5

 Arc 2: f3 - 5 g2 <=  6

 Arc 3: f1 - 5 g3 <=  2

 Arc 4: f2 - 5 g4 <=  4

 Arc 5: f3 - 5 g5 <=  5

 Arc 6: f1 - 5 g6 <=  4

 Arc 7: f2 + f3 - 5 g7 <=  7

 Budget: 250 g1 + 300 g2 + 100 g3 + 200 g4 + 250 g5 +
         200 g6 + 350 g7  <= 1000

* ------ end ------ *


Statistics-
  MIP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  280K bytes.
  Objective Function is MAXIMIZED.
  MIP Strategy:      1
  Variables:        10
    Integer:         7
  Constraints:       8
```

```
    8 LE,   0 EQ,   0 GE.
  Non-zero LP elements:      23
  Disk Space:      0K bytes.
  Page Space:      1K bytes.
  Capacity:  3.3% used.
  Estimated Time: 00:00:00

Iter    7
Solution Time:  00:00:00
U N I Q U E   S O L U T I O N

Optimal Solution:      15.0182 Max Node Depth:    656 Limit: NONE

Solution: 13.3000 Iter:      30 Nodes:       7 Iteration Time: 00:00:03
INTEGER SOLUTION
```

File: n1imp2 11/19/88 00:31:59 Page 1-1

SOLUTION (Maximized): 13.3000 Investment Strategy Model 2

Variable	Activity	Cost	Variable	Activity	Cost
f1	7.0000	0.9000	f2	7.0000	1.0000
f3	0.0000	0.0500	g1	2.0000	0.0000
g2	0.0000	0.0000	g3	1.0000	0.0000
g4	1.0000	0.0000	g5	0.0000	0.0000
g6	1.0000	0.0000	g7	0.0000	0.0000

File: n1imp2 11/19/88 00:31:59 Page 1-2

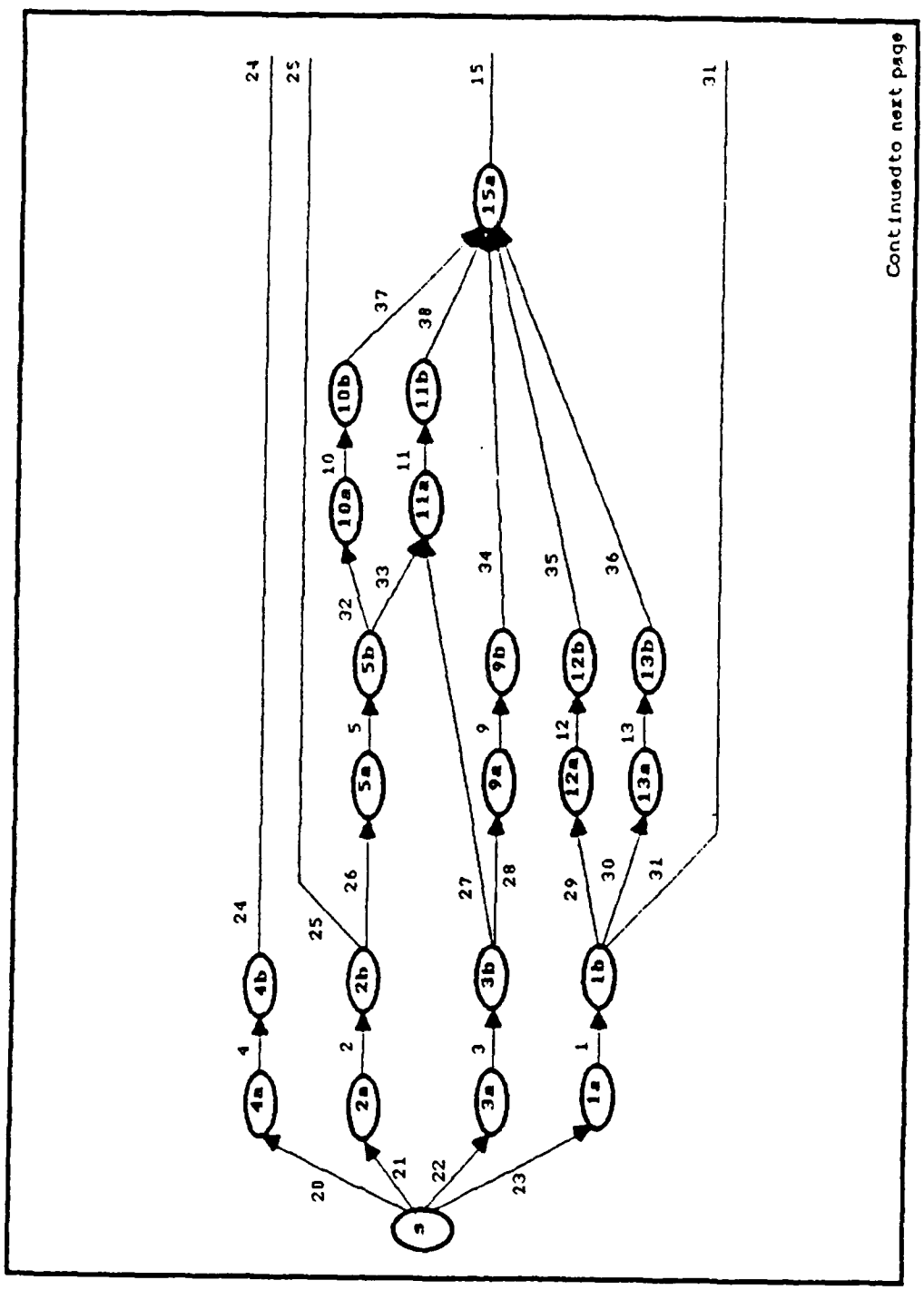
CONSTRAINTS: Investment Strategy Model 2

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 1	4.0000 <	5.0000	Arc 2	0.0000 <	6.0000
Arc 3	2.0000 <	2.0000	Arc 4	2.0000 <	4.0000
Arc 5	0.0000 <	5.0000	Arc 6	2.0000 <	4.0000
Arc 7	7.0000 <	7.0000	Budget	1,000.0000 <	1,000.0000

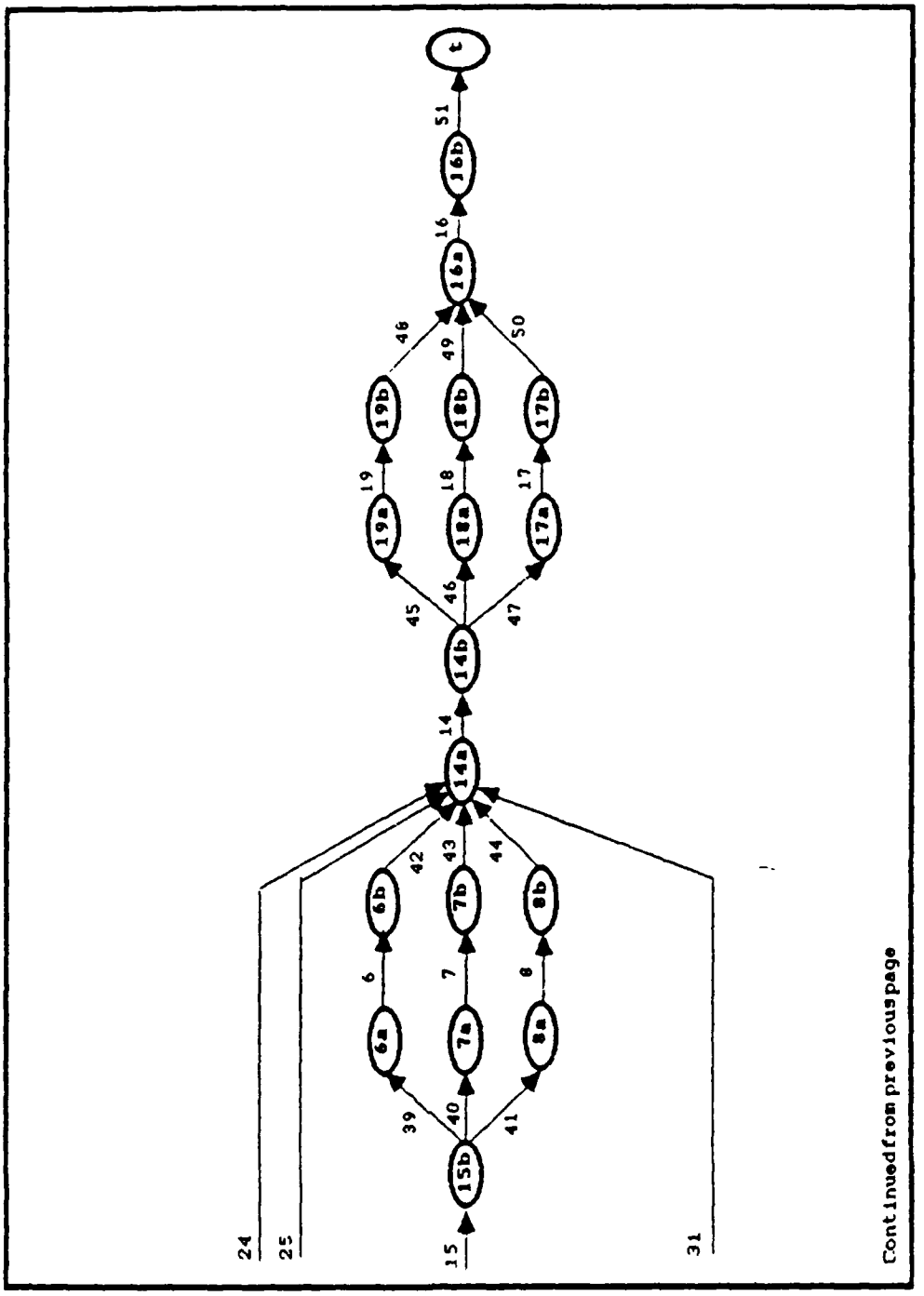
Total Error: 0.000000

Appendix D. *Experimental Results for Network A*

D.1 Revised Topology of Network A



Revised Topology Of Network A



Revised Topology Of Network A

D.2 Input File to FORMULA

```
/*****************************************************************/
/*                          Network A                            */
/*****************************************************************/

% Arc relationship

 arc(s,20).
 arc(s,21).
 arc(s,22).
 arc(s,23).
 arc(1,29).
 arc(1,30).
 arc(1,31).
 arc(2,25).
 arc(2,26).
 arc(3,27).
 arc(3,28).
 arc(4,24).
 arc(5,32).
 arc(5,33).
 arc(6,42).
 arc(7,43).
 arc(8,44).
 arc(9,34).
 arc(10,37).
 arc(11,38).
 arc(12,35).
 arc(13,36).
 arc(14,45).
 arc(14,46).
 arc(14,47).
 arc(15,39).
 arc(15,40).
 arc(15,41).
 arc(16,51).
 arc(17,50).
 arc(18,49).
 arc(19,48).
 arc(20,4).
 arc(21,2).
 arc(22,3).
 arc(23,1).
 arc(24,14).
 arc(25,14).
 arc(26,5).
 arc(27,11).
 arc(28,9).
 arc(29,12).
 arc(30,13).
```

```
arc(31,14).
arc(32,10).
arc(33,11).
arc(34,15).
arc(35,15).
arc(36,15).
arc(37,15).
arc(38,15).
arc(39,6).
arc(40,7).
arc(41,8).
arc(42,14).
arc(43,14).
arc(44,14).
arc(45,19).
arc(46,18).
arc(47,17).
arc(48,16).
arc(49,16).
arc(50,16).
arc(51,t).

% Component survival probability

prob(s,1).
prob(1,1).
prob(2,0.3).
prob(3,0.7).
prob(4,0.5).
prob(5,0.8).
prob(6,1).
prob(7,0.3).
prob(8,0.7).
prob(9,0.5).
prob(10,0.8).
prob(11,1).
prob(12,0.3).
prob(13,0.7).
prob(14,0.7).
prob(15,0.8).
prob(16,0.8).
prob(17,0.7).
prob(18,0.3).
prob(19,1).
prob(20,1).
prob(21,1).
prob(22,1).
prob(23,1).
prob(24,1).
prob(25,0.6).
prob(26,0.3).
```

```
prob(27,1).
prob(28,1).
prob(29,1).
prob(30,1).
prob(31,1).
prob(32,0.6).
prob(33,0.7).
prob(34,1).
prob(35,0.7).
prob(36,1).
prob(37,0.6).
prob(38,1).
prob(39,0.3).
prob(40,0.6).
prob(41,0.7).
prob(42,0.6).
prob(43,0.6).
prob(44,0.3).
prob(45,0.6).
prob(46,0.6).
prob(47,0.3).
prob(48,0.3).
prob(49,0.6).
prob(50,0.7).
prob(51,1).
prob(t,1).

% Component capacity

cap(1,*).
cap(2,*).
cap(3,*).
cap(4,*).
cap(5,*).
cap(6,*).
cap(7,*).
cap(8,*).
cap(9,*).
cap(10,*).
cap(11,*).
cap(12,*).
cap(13,*).
cap(14,*).
cap(15,*).
cap(16,*).
cap(17,*).
cap(18,*).
cap(19,*).
cap(20,*).
cap(21,*).
cap(22,*).
```

```
 cap(23,*).
 cap(24,1200).
 cap(25,1200).
 cap(26,1200).
 cap(27,1200).
 cap(28,1200).
 cap(29,1200).
 cap(30,1200).
 cap(31,1200).
 cap(32,1200).
 cap(33,1200).
 cap(34,4800).
 cap(35,4800).
 cap(36,4800).
 cap(37,4800).
 cap(38,4800).
 cap(39,4800).
 cap(40,4800).
 cap(41,4800).
 cap(42,4800).
 cap(43,4800).
 cap(44,4800).
 cap(45,4800).
 cap(46,4800).
 cap(47,4800).
 cap(48,4800).
 cap(49,4800).
 cap(50,4800).
 cap(51,*).

% Cost of increasing capacity by one unit

 cost(_,100).

% Predetermined lump sum of capacity increase

 invest(_,100).

% Budget available

 budget(100000).

/*-------- end --------*/
```

D.3 A Listing of Paths and Path Reliabilities

```
**********************************************************
*  Following is a list of all paths from "s" to "t"     *
*  of the network described in the input data file.      *
**********************************************************

  Path 1: s 20 4 24 14 45 19 48 16 51 t
  Reliability: 0.036

  Path 2: s 20 4 24 14 46 18 49 16 51 t
  Reliability: 0.0216

  Path 3: s 20 4 24 14 47 17 50 16 51 t
  Reliability: 0.0294

  Path 4: s 21 2 25 14 45 19 48 16 51 t
  Reliability: 0.01296

  Path 5: s 21 2 25 14 46 18 49 16 51 t
  Reliability: 0.007776

  Path 6: s 21 2 25 14 47 17 50 16 51 t
  Reliability: 0.010584

  Path 7: s 21 2 26 5 32 10 37 15 39 6 42 14 45 19 48 16 51 t
  Reliability: 0.00021499

  Path 8: s 21 2 26 5 32 10 37 15 39 6 42 14 46 18 49 16 51 t
  Reliability: 0.00012899

  Path 9: s 21 2 26 5 32 10 37 15 39 6 42 14 47 17 50 16 51 t
  Reliability: 0.00017558

  Path 10: s 21 2 26 5 32 10 37 15 40 7 43 14 45 19 48 16 51 t
  Reliability: 0.00012899

  Path 11: s 21 2 26 5 32 10 37 15 40 7 43 14 46 18 49 16 51 t
  Reliability: 0.0000774

  Path 12: s 21 2 26 5 32 10 37 15 40 7 43 14 47 17 50 16 51 t
  Reliability: 0.00010535

  Path 13: s 21 2 26 5 32 10 37 15 41 8 44 14 45 19 48 16 51 t
  Reliability: 0.00017558

  Path 14: s 21 2 26 5 32 10 37 15 41 8 44 14 46 18 49 16 51 t
  Reliability: 0.00010535

  Path 15: s 21 2 26 5 32 10 37 15 41 8 44 14 47 17 50 16 51 t
  Reliability: 0.00014339
```

Path 16: s 21 2 26 5 33 11 38 15 39 6 42 14 45 19 48 16 51 t
Reliability: 0.00052255

Path 17: s 21 2 26 5 33 11 38 15 39 6 42 14 46 18 49 16 51 t
Reliability: 0.00031353

Path 18: s 21 2 26 5 33 11 38 15 39 6 42 14 47 17 50 16 51 t
Reliability: 0.00042675

Path 19: s 21 2 26 5 33 11 38 15 40 7 43 14 45 19 48 16 51 t
Reliability: 0.00031353

Path 20: s 21 2 26 5 33 11 38 15 40 7 43 14 46 18 49 16 51 t
Reliability: 0.00018812

Path 21: s 21 2 26 5 33 11 38 15 40 7 43 14 47 17 50 16 51 t
Reliability: 0.00025605

Path 22: s 21 2 26 5 33 11 38 15 41 8 44 14 45 19 48 16 51 t
Reliability: 0.00042675

Path 23: s 21 2 26 5 33 11 38 15 41 8 44 14 46 18 49 16 51 t
Reliability: 0.00025605

Path 24: s 21 2 26 5 33 11 38 15 41 8 44 14 47 17 50 16 51 t
Reliability: 0.00034851

Path 25: s 22 3 27 11 38 15 39 6 42 14 45 19 48 16 51 t
Reliability: 0.0072576

Path 26: s 22 3 27 11 38 15 39 6 42 14 46 18 49 16 51 t
Reliability: 0.00435456

Path 27: s 22 3 27 11 38 15 39 6 42 14 47 17 50 16 51 t
Reliability: 0.00592704

Path 28: s 22 3 27 11 38 15 40 7 43 14 45 19 48 16 51 t
Reliability: 0.00435456

Path 29: s 22 3 27 11 38 15 40 7 43 14 46 18 49 16 51 t
Reliability: 0.00261274

Path 30: s 22 3 27 11 38 15 40 7 43 14 47 17 50 16 51 t
Reliability: 0.00355622

Path 31: s 22 3 27 11 38 15 41 8 44 14 45 19 48 16 51 t
Reliability: 0.00592704

Path 32: s 22 3 27 11 38 15 41 8 44 14 46 18 49 16 51 t
Reliability: 0.00355622

Path 33: s 22 3 27 11 38 15 41 8 44 14 47 17 50 16 51 t
Reliability: 0.00484042

Path 34: s 22 3 28 9 34 15 39 6 42 14 45 19 48 16 51 t
Reliability: 0.0036288

Path 35: s 22 3 28 9 34 15 39 6 42 14 46 18 49 16 51 t
Reliability: 0.00217728

Path 36: s 22 3 28 9 34 15 39 6 42 14 47 17 50 16 51 t
Reliability: 0.00296352

Path 37: s 22 3 28 9 34 15 40 7 43 14 45 19 48 16 51 t
Reliability: 0.00217728

Path 38: s 22 3 28 9 34 15 40 7 43 14 46 18 49 16 51 t
Reliability: 0.00130637

Path 39: s 22 3 28 9 34 15 40 7 43 14 47 17 50 16 51 t
Reliability: 0.00177811

Path 40: s 22 3 28 9 34 15 41 8 44 14 45 19 48 16 51 t
Reliability: 0.00296352

Path 41: s 22 3 28 9 34 15 41 8 44 14 46 18 49 16 51 t
Reliability: 0.00177811

Path 42: s 22 3 28 9 34 15 41 8 44 14 47 17 50 16 51 t
Reliability: 0.00242021

Path 43: s 23 1 29 12 35 15 39 6 42 14 45 19 48 16 51 t
Reliability: 0.00217728

Path 44: s 23 1 29 12 35 15 39 6 42 14 46 18 49 16 51 t
Reliability: 0.00130637

Path 45: s 23 1 29 12 35 15 39 6 42 14 47 17 50 16 51 t
Reliability: 0.00177811

Path 46: s 23 1 29 12 35 15 40 7 43 14 45 19 48 16 51 t
Reliability: 0.00130637

Path 47: s 23 1 29 12 35 15 40 7 43 14 46 18 49 16 51 t
Reliability: 0.00078382

Path 48: s 23 1 29 12 35 15 40 7 43 14 47 17 50 16 51 t
Reliability: 0.00106687

Path 49: s 23 1 29 12 35 15 41 8 44 14 45 19 48 16 51 t
Reliability: 0.00177811

```
Path 50: s 23 1 29 12 35 15 41 8 44 14 46 18 49 16 51 t
Reliability: 0.00106687

Path 51: s 23 1 29 12 35 15 41 8 44 14 47 17 50 16 51 t
Reliability: 0.00145212

Path 52: s 23 1 30 13 36 15 39 6 42 14 45 19 48 16 51 t
Reliability: 0.0072576

Path 53: s 23 1 30 13 36 15 39 6 42 14 46 18 49 16 51 t
Reliability: 0.00435456

Path 54: s 23 1 30 13 36 15 39 6 42 14 47 17 50 16 51 t
Reliability: 0.00592704

Path 55: s 23 1 30 13 36 15 40 7 43 14 45 19 48 16 51 t
Reliability: 0.00435456

Path 56: s 23 1 30 13 36 15 40 7 43 14 46 18 49 16 51 t
Reliability: 0.00261274

Path 57: s 23 1 30 13 36 15 40 7 43 14 47 17 50 16 51 t
Reliability: 0.00355622

Path 58: s 23 1 30 13 36 15 41 8 44 14 45 19 48 16 51 t
Reliability: 0.00592704

Path 59: s 23 1 30 13 36 15 41 8 44 14 46 18 49 16 51 t
Reliability: 0.00355622

Path 60: s 23 1 30 13 36 15 41 8 44 14 47 17 50 16 51 t
Reliability: 0.00484042

Path 61: s 23 1 31 14 45 19 48 16 51 t
Reliability: 0.072

Path 62: s 23 1 31 14 46 18 49 16 51 t
Reliability: 0.0432

Path 63: s 23 1 31 14 47 17 50 16 51 t
Reliability: 0.0588

* ------ end ------ *
```

D.4 Solution to Maximum Flow Model

```
LP83 a:amax2 output a:amax2out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Maximum Flow Formulation

 ..Objective Maximize

  f1 +  f2 +  f3 +  f4 +
  f5 +  f6 +  f7 +  f8 +
  f9 +  f10 +  f11 +  f12 +
  f13 +  f14 +  f15 +  f16 +
  f17 +  f18 +  f19 +  f20 +
  f21 +  f22 +  f23 +  f24 +
  f25 +  f26 +  f27 +  f28 +
  f29 +  f30 +  f31 +  f32 +
  f33 +  f34 +  f35 +  f36 +
  f37 +  f38 +  f39 +  f40 +
  f41 +  f42 +  f43 +  f44 +
  f45 +  f46 +  f47 +  f48 +
  f49 +  f50 +  f51 +  f52 +
  f53 +  f54 +  f55 +  f56 +
  f57 +  f58 +  f59 +  f60 +
  f61 +  f62 +  f63

 ..Constraints

 Arc 24: f1 + f2 + f3 <=  1200

 Arc 25: f4 + f5 + f6 <=  1200

 Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
         f14 + f15 + f16 + f17 + f18 + f19 + f20 +
         f21 + f22 + f23 + f24 <=  1200

 Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
         f32 + f33 <=  1200

 Arc 28: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
         f41 + f42 <=  1200

 Arc 29: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
```

```
        f50 + f51 <=  1200

Arc 30: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  1200

Arc 31: f61 + f62 + f63 <=  1200

Arc 32: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  1200

Arc 33: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 <=  1200

Arc 34: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 <=  4800

Arc 35: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 <=  4800

Arc 36: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  4800

Arc 37: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  4800

Arc 38: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 + f25 + f26 + f27 + f28 + f29 +
        f30 + f31 + f32 + f33 <=  4800

Arc 39: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  4800

Arc 40: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  4800

Arc 41: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 <=  4800

Arc 42: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  4800

Arc 43: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  4800

Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
```

```
        f51 + f58 + f59 + f60 <=  4800

 Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  4800

 Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  4800

 Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  4800

 Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  4800

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  4800

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  4800

* ------ end ------ *



Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  398K bytes.
  Objective Function is MAXIMIZED.
  Variables:         63
  Constraints:       27
    27 LE,    0 EQ,    0 GE.
  Non-zero LP elements:      369
  Disk Space:      0K bytes.
  Page Space:     14K bytes.
  Capacity:  9.8% used.
  Estimated Time: 00:01:01

Iter   12
Solution Time:  00:00:02
A L T E R N A T E   S O L U T I O N S


File: amax2                                    11/25/88 18:44:35 Page 1-1
```

SOLUTION (Maximized): 9,600.0000 Maximum Flow Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f1	0.0000	1.0000	f2	1,200.0000	1.0000
f3	0.0000	1.0000	f4	0.0000	1.0000
f5	1,200.0000	1.0000	f6	0.0000	1.0000
f7	1,200.0000	1.0000	f8	0.0000	1.0000
f9	0.0000	1.0000	f10	0.0000	1.0000
f11	0.0000	1.0000	f12	0.0000	1.0000
f13	0.0000	1.0000	f14	0.0000	1.0000
f15	0.0000	1.0000	f16	0.0000	1.0000
f17	0.0000	1.0000	f18	0.0000	1.0000
f19	0.0000	1.0000	f20	0.0000	1.0000

SOLUTION (Maximized): 9,600.0000 Maximum Flow Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f21	0.0000	1.0000	f22	0.0000	1.0000
f23	0.0000	1.0000	f24	0.0000	1.0000
f25	1,200.0000	1.0000	f26	0.0000	1.0000
f27	0.0000	1.0000	f28	0.0000	1.0000
f29	0.0000	1.0000	f30	0.0000	1.0000
f31	0.0000	1.0000	f32	0.0000	1.0000
f33	0.0000	1.0000	f34	1,200.0000	1.0000
f35	0.0000	1.0000	f36	0.0000	1.0000
f37	0.0000	1.0000	f38	0.0000	1.0000
f39	0.0000	1.0000	f40	0.0000	1.0000

SOLUTION (Maximized): 9,600.0000 Maximum Flow Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	1.0000	f42	0.0000	1.0000
f43	0.0000	1.0000	f44	1,200.0000	1.0000
f45	0.0000	1.0000	f46	0.0000	1.0000
f47	0.0000	1.0000	f48	0.0000	1.0000
f49	0.0000	1.0000	f50	0.0000	1.0000
f51	0.0000	1.0000	f52	0.0000	1.0000
f53	0.0000	1.0000	f54	0.0000	1.0000
f55	0.0000	1.0000	f56	1,200.0000	1.0000
f57	0.0000	1.0000	f58	0.0000	1.0000
f59	0.0000	1.0000	f60	0.0000	1.0000

File: amax2 11/25/88 18:44:35 Page 1-4
SOLUTION (Maximized): 9,600.0000 Maximum Flow Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f61	1,200.0000	1.0000	f62	0.0000	1.0000
f63	0.0000	1.0000			

File: amax2 11/25/88 18:44:35 Page 1-5
CONSTRAINTS: Maximum Flow Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 24	1,200.0000 <	1,200.0000	Arc 25	1,200.0000 <	1,200.0000
Arc 26	1,200.0000 <	1,200.0000	Arc 27	1,200.0000 <	1,200.0000
Arc 28	1,200.0000 <	1,200.0000	Arc 29	1,200.0000 <	1,200.0000
Arc 30	1,200.0000 <	1,200.0000	Arc 31	1,200.0000 <	1,200.0000
Arc 32	1,200.0000 <	1,200.0000	Arc 33	0.0000 <	1,200.0000
Arc 34	1,200.0000 <	4,800.0000	Arc 35	1,200.0000 <	4,800.0000

I	Arc 36	1,200.0000 < 4,800.0000	I	Arc 37	1,200.0000 < 4,800.0000
I	Arc 38	1,200.0000 < 4,800.0000	I	Arc 39	4,800.0000 < 4,800.0000
I	Arc 40	1,200.0000 < 4,800.0000	I	Arc 41	0.0000 < 4,800.0000
I	Arc 42	4,800.0000 < 4,800.0000	I	Arc 43	1,200.0000 < 4,800.0000

File: amax2 11/25/88 18:44:35 Page 1-6

CONSTRAINTS: Maximum Flow Formulation

I	Arc 44	0.0000 < 4,800.0000	I	Arc 45	4,800.0000 < 4,800.0000
I	Arc 46	4,800.0000 < 4,800.0000	I	Arc 47	0.0000 < 4,800.0000
I	Arc 48	4,800.0000 < 4,800.0000	I	Arc 49	4,800.0000 < 4,800.0000
I	Arc 50	0.0000 < 4,800.0000			

Total Error: 0.000000

D.5 Solution to Lower Bound Model

```
LP83 a:alow output a:alowout alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Lower Bound Formulation

 ..Objective Maximize

 0.036 f1 + 0.0216 f2 + 0.0294 f3 + 0.01296 f4 +
 0.007776 f5 + 0.010584 f6 + 0.00021499 f7 + 0.00012899 f8 +
 0.00017558 f9 + 0.00012899 f10 + 0.0000774 f11 + 0.00010535 f12 +
 0.00017558 f13 + 0.00010535 f14 + 0.00014339 f15 + 0.00052255 f16 +
 0.00031353 f17 + 0.00042675 f18 + 0.00031353 f19 + 0.00018812 f20 +
 0.00025605 f21 + 0.00042675 f22 + 0.00025605 f23 + 0.00034851 f24 +
 0.0072576 f25 + 0.00435456 f26 + 0.00592704 f27 + 0.00435456 f28 +
 0.00261274 f29 + 0.00355622 f30 + 0.00592704 f31 + 0.00355622 f32 +
 0.00484042 f33 + 0.0036288 f34 + 0.00217728 f35 + 0.00296352 f36 +
 0.00217728 f37 + 0.00130637 f38 + 0.00177811 f39 + 0.00296352 f40 +
 0.00177811 f41 + 0.00242021 f42 + 0.00217728 f43 + 0.00130637 f44 +
 0.00177811 f45 + 0.00130637 f46 + 0.00078382 f47 + 0.00106687 f48 +
 0.00177811 f49 + 0.00106687 f50 + 0.00145212 f51 + 0.0072576 f52 +
 0.00435456 f53 + 0.00592704 f54 + 0.00435456 f55 + 0.00261274 f56 +
 0.00355622 f57 + 0.00592704 f58 + 0.00355622 f59 + 0.00484042 f60 +
 0.072 f61 + 0.0432 f62 + 0.0588 f63

 ..Constraints

 Arc 24: f1 + f2 + f3 <=  1200

 Arc 25: f4 + f5 + f6 <=  1200

 Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
         f14 + f15 + f16 + f17 + f18 + f19 + f20 +
         f21 + f22 + f23 + f24 <=  1200

 Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
         f32 + f33 <=  1200

 Arc 28: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
         f41 + f42 <=  1200

 Arc 29: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
```

```
        f50 + f51 <=  1200

Arc 30: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  1200

Arc 31: f61 + f62 + f63 <=  1200

Arc 32: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  1200

Arc 33: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 <=  1200

Arc 34: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 <=  4800

Arc 35: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 <=  4800

Arc 36: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  4800

Arc 37: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  4800

Arc 38: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 + f25 + f26 + f27 + f28 + f29 +
        f30 + f31 + f32 + f33 <=  4800

Arc 39: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  4800

Arc 40: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  4800

Arc 41: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 <=  4800

Arc 42: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  4800

Arc 43: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  4800

Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
```

```
        f51 + f58 + f59 + f60 <=  4800

 Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  4800

 Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  4800

 Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  4800

 Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  4800

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  4800

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  4800

* ------ end ------ *



Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  398K bytes.
  Objective Function is MAXIMIZED.
  Variables:        63
  Constraints:      27
    27 LE,   0 EQ,   0 GE.
  Non-zero LP elements:     369
  Disk Space:     0K bytes.
  Page Space:    14K bytes.
  Capacity:  9.8% used.
  Estimated Time: 00:01:01

Iter   12
Solution Time:  00:00:02
A L T E R N A T E   S O L U T I O N S


File: alow                                        11/25/88 18:45:22 Page 1-1
```

SOLUTION (Maximized): 167.0817 Lower Bound Formulation

	Variable	Activity	Cost		Variable	Activity	Cost
I	f1	1,200.0000	0.0360		f2	0.0000	0.0216
	f3	0.0000	0.0294	I	f4	1,200.0000	0.0130
	f5	0.0000	0.0078		f6	0.0000	0.0106
	f7	0.0000	0.0002		f8	0.0000	0.0001
	f9	0.0000	0.0002		f10	0.0000	0.0001
	f11	0.0000	0.0001		f12	0.0000	0.0001
	f13	0.0000	0.0002	I	f14	0.0000	0.0001
	f15	0.0000	0.0001		f16	0.0000	0.0005
	f17	0.0000	0.0003		f18	0.0000	0.0004
	f19	0.0000	0.0003		f20	0.0000	0.0002

SOLUTION (Maximized): 167.0817 Lower Bound Formulation

	Variable	Activity	Cost		Variable	Activity	Cost
	f21	0.0000	0.0003		f22	0.0000	0.0004
I	f23	0.0000	0.0003	I	f24	1,200.0000	0.0003
I	f25	1,200.0000	0.0073		f26	0.0000	0.0044
	f27	0.0000	0.0059		f28	0.0000	0.0044
	f29	0.0000	0.0026		f30	0.0000	0.0036
	f31	0.0000	0.0059		f32	0.0000	0.0036
	f33	0.0000	0.0048		f34	0.0000	0.0036
	f35	0.0000	0.0022	I	f36	1,200.0000	0.0030
	f37	0.0000	0.0022		f38	0.0000	0.0013
	f39	0.0000	0.0018		f40	0.0000	0.0030

SOLUTION (Maximized): 167.0817 Lower Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	0.0018	f42	0.0000	0.0024
f43	0.0000	0.0022	f44	0.0000	0.0013
f45	1,200.0000	0.0018	f46	0.0000	0.0013
f47	0.0000	0.0008	f48	0.0000	0.0011
f49	0.0000	0.0018	f50	0.0000	0.0011
f51	0.0000	0.0015	f52	0.0000	0.0073
f53	0.0000	0.0044	f54	1,200.0000	0.0059
f55	0.0000	0.0044	f56	0.0000	0.0026
f57	0.0000	0.0036	f58	0.0000	0.0059
f59	0.0000	0.0036	f60	0.0000	0.0048

File: alow 11/25/88 18:45:22 Page 1-4
SOLUTION (Maximized): 167.0817 Lower Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f61	1,200.0000	0.0720	f62	0.0000	0.0432
f63	0.0000	0.0588			

File: alow 11/25/88 18:45:22 Page 1-5
CONSTRAINTS: Lower Bound Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 24	1,200.0000	< 1,200.0000	Arc 25	1,200.0000	< 1,200.0000
Arc 26	1,200.0000	< 1,200.0000	Arc 27	1,200.0000	< 1,200.0000
Arc 28	1,200.0000	< 1,200.0000	Arc 29	1,200.0000	< 1,200.0000
Arc 30	1,200.0000	< 1,200.0000	Arc 31	1,200.0000	< 1,200.0000
Arc 32	0.0000	< 1,200.0000	Arc 33	1,200.0000	< 1,200.0000
Arc 34	1,200.0000	< 4,800.0000	Arc 35	1,200.0000	< 4,800.0000

I	Arc 36	1,200.0000 < 4,800.0000 I	Arc 37	0.0000 < 4,800.0000	
I	Arc 38	2,400.0000 < 4,800.0000 I	Arc 39	4,800.0000 < 4,800.0000	
I	Arc 40	0.0000 < 4,800.0000 I	Arc 41	1,200.0000 < 4,800.0000	
	Arc 42	4,800.0000 < 4,800.0000 I	Arc 43	0.0000 < 4,800.0000	

File: alow 11/25/88 18:45:22 Page 1-6

CONSTRAINTS: Lower Bound Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
I Arc 44	1,200.0000 < 4,800.0000 I		Arc 45	4,800.0000 < 4,800.0000	
I Arc 46	0.0000 < 4,800.0000 I		Arc 47	4,800.0000 < 4,800.0000	
Arc 48	4,800.0000 < 4,800.0000 I		Arc 49	0.0000 < 4,800.0000	
Arc 50	4,800.0000 < 4,800.0000				

Total Error: 0.000000

D.6 Solution to Upper Bound Model

```
LP83 a:aup2 output a:aup2out alternate 1

 Copyright (C) 1986 by Sunset Software.
 All Rights Reserved Worldwide.
 1613 Chelsea Road,  Suite 153
 San Marino, California  91108   U.S.A.

 Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Upper Bound Formulation

 ..Objective Maximize

  f1 +  f2 +  f3 +  f4 +
  f5 +  f6 +  f7 +  f8 +
  f9 +  f10 +  f11 +  f12 +
  f13 +  f14 +  f15 +  f16 +
  f17 +  f18 +  f19 +  f20 +
  f21 +  f22 +  f23 +  f24 +
  f25 +  f26 +  f27 +  f28 +
  f29 +  f30 +  f31 +  f32 +
  f33 +  f34 +  f35 +  f36 +
  f37 +  f38 +  f39 +  f40 +
  f41 +  f42 +  f43 +  f44 +
  f45 +  f46 +  f47 +  f48 +
  f49 +  f50 +  f51 +  f52 +
  f53 +  f54 +  f55 +  f56 +
  f57 +  f58 +  f59 +  f60 +
  f61 +  f62 +  f63

 ..Constraints

 Arc 24: f1 + f2 + f3 <=  1200

 Arc 25: f4 + f5 + f6 <=  720.0

 Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
         f14 + f15 + f16 + f17 + f18 + f19 + f20 +
         f21 + f22 + f23 + f24 <=  360.0

 Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
         f32 + f33 <=  1200

 Arc 28: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
         f41 + f42 <=  1200

 Arc 29: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
```

```
        f50 + f51 <=  1200

Arc 30: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  1200

Arc 31: f61 + f62 + f63 <=  1200

Arc 32: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  720.0

Arc 33: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 <=  840.0

Arc 34: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 <=  4800

Arc 35: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 <=  3360.0

Arc 36: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 <=  4800

Arc 37: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 <=  2880.0

Arc 38: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 + f25 + f26 + f27 + f28 + f29 +
        f30 + f31 + f32 + f33 <=  4800

Arc 39: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  1440.0

Arc 40: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  2880.0

Arc 41: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 <=  3360.0

Arc 42: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 <=  2880.0

Arc 43: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 <=  2880.0

Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
```

```
        f51 + f58 + f59 + f60 <=  1440.0

 Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  2880.0

 Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  2880.0

 Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  1440.0

 Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 <=  1440.0

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 <=  2880.0

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 <=  3360.0

* ------ end ------ *



Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  398K bytes.
  Objective Function is MAXIMIZED.
  Variables:        63
  Constraints:      27
    27 LE,   0 EQ,   0 GE.
  Non-zero LP elements:     369
  Disk Space:      0K bytes.
  Page Space:     14K bytes.
  Capacity:  9.8% used.
  Estimated Time: 00:01:01

Iter    9
Solution Time:  00:00:02
A L T E R N A T E   S O L U T I O N S


File: aup2                                        11/25/88 18:46:11 Page 1-1
```

SOLUTION (Maximized): 5,760.0000 Upper Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f1	0.0000	1.0000	f2	1,200.0000	1.0000
f3	0.0000	1.0000	f4	0.0000	1.0000
f5	720.0000	1.0000	f6	0.0000	1.0000
f7	120.0000	1.0000	f8	120.0000	1.0000
f9	0.0000	1.0000	f10	120.0000	1.0000
f11	0.0000	1.0000	f12	0.0000	1.0000
f13	0.0000	1.0000	f14	0.0000	1.0000
f15	0.0000	1.0000	f16	0.0000	1.0000
f17	0.0000	1.0000	f18	0.0000	1.0000
f19	0.0000	1.0000	f20	0.0000	1.0000

SOLUTION (Maximized): 5,760.0000 Upper Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f21	0.0000	1.0000	f22	0.0000	1.0000
f23	0.0000	1.0000	f24	0.0000	1.0000
f25	0.0000	1.0000	f26	840.0000	1.0000
f27	360.0000	1.0000	f28	0.0000	1.0000
f29	0.0000	1.0000	f30	0.0000	1.0000
f31	0.0000	1.0000	f32	0.0000	1.0000
f33	0.0000	1.0000	f34	0.0000	1.0000
f35	0.0000	1.0000	f36	0.0000	1.0000
f37	0.0000	1.0000	f38	0.0000	1.0000
f39	1,080.0000	1.0000	f40	0.0000	1.0000

SOLUTION (Maximized): 5,760.0000 Upper Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	1.0000	f42	0.0000	1.0000
f43	0.0000	1.0000	f44	0.0000	1.0000
f45	0.0000	1.0000	f46	0.0000	1.0000
f47	0.0000	1.0000	f48	0.0000	1.0000
f49	0.0000	1.0000	f50	0.0000	1.0000
f51	0.0000	1.0000	f52	0.0000	1.0000
f53	0.0000	1.0000	f54	0.0000	1.0000
f55	0.0000	1.0000	f56	0.0000	1.0000
f57	0.0000	1.0000	f58	0.0000	1.0000
f59	0.0000	1.0000	f60	0.0000	1.0000

File: aup2 11/25/88 18:46:11 Page 1-4
SOLUTION (Maximized): 5,760.0000 Upper Bound Formulation

Variable	Activity	Cost	Variable	Activity	Cost
I f61	1,200.0000	1.0000	f62	0.0000	1.0000
f63	0.0000	1.0000			

File: aup2 11/25/88 18:46:11 Page 1-5
CONSTRAINTS: Upper Bound Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 24	1,200.0000 <	1,200.0000	Arc 25	720.0000 <	720.0000
Arc 26	360.0000 <	360.0000	Arc 27	1,200.0000 <	1,200.0000
I Arc 28	1,080.0000 <	1,200.0000	I Arc 29	0.0000 <	1,200.0000
I Arc 30	0.0000 <	1,200.0000	Arc 31	1,200.0000 <	1,200.0000
I Arc 32	360.0000 <	720.0000	I Arc 33	0.0000 <	840.0000

I	Arc 34	1,080.0000 < 4,800.0000	I	Arc 35	0.0000 < 3,360.0000
I	Arc 36	0.0000 < 4,800.0000	I	Arc 37	360.0000 < 2,880.0000
I	Arc 38	1,200.0000 < 4,800.0000	I	Arc 39	1,440.0000 < 1,440.0000
I	Arc 40	1,200.0000 < 2,880.0000	I	Arc 41	0.0000 < 3,360.0000
I	Arc 42	1,440.0000 < 2,880.0000	I	Arc 43	1,200.0000 < 2,880.0000

CONSTRAINTS: Upper Bound Formulation

Constraint	Activity	RHS	Constraint	Activity	RHS
I	Arc 44	0.0000 < 1,440.0000	I	Arc 45	1,440.0000 < 2,880.0000
I	Arc 46	2,880.0000 < 2,880.0000	I	Arc 47	1,440.0000 < 1,440.0000
I	Arc 48	1,440.0000 < 1,440.0000	I	Arc 49	2,880.0000 < 2,880.0000
I	Arc 50	1,440.0000 < 3,360.0000			

Total Error: 0.000000

D.7 Solution to Investment Strategy Model 1 (Case 1)

```
LP83 a:aimp11 output a:aimp11out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Investment Strategy Model 1

 ..Objective Maximize

 0.036 f1 + 0.0216 f2 + 0.0294 f3 + 0.01296 f4 +
 0.007776 f5 + 0.010584 f6 + 0.00021499 f7 + 0.00012899 f8 +
 0.00017558 f9 + 0.00012899 f10 + 0.0000774 f11 + 0.00010535 f12 +
 0.00017558 f13 + 0.00010535 f14 + 0.00014339 f15 + 0.00052255 f16 +
 0.00031353 f17 + 0.00042675 f18 + 0.00031353 f19 + 0.00018812 f20 +
 0.00025605 f21 + 0.00042675 f22 + 0.00025605 f23 + 0.00034851 f24 +
 0.0072576 f25 + 0.00435456 f26 + 0.00592704 f27 + 0.00435456 f28 +
 0.00261274 f29 + 0.00355622 f30 + 0.00592704 f31 + 0.00355622 f32 +
 0.00484042 f33 + 0.0036288 f34 + 0.00217728 f35 + 0.00296352 f36 +
 0.00217728 f37 + 0.00130637 f38 + 0.00177811 f39 + 0.00296352 f40 +
 0.00177811 f41 + 0.00242021 f42 + 0.00217728 f43 + 0.00130637 f44 +
 0.00177811 f45 + 0.00130637 f46 + 0.00078382 f47 + 0.00106687 f48 +
 0.00177811 f49 + 0.00106687 f50 + 0.00145212 f51 + 0.0072576 f52 +
 0.00435456 f53 + 0.00592704 f54 + 0.00435456 f55 + 0.00261274 f56 +
 0.00355622 f57 + 0.00592704 f58 + 0.00355622 f59 + 0.00484042 f60 +
 0.072 f61 + 0.0432 f62 + 0.0588 f63 +
 0 d24 + 0 d25 + 0 d26 + 0 d27 + 0 d28 + 0 d29 + 0 d30 +
 0 d31 + 0 d32 + 0 d33 + 0 d34 + 0 d35 + 0 d36 + 0 d37 +
 0 d38 + 0 d39 + 0 d40 + 0 d41 + 0 d42 + 0 d43 + 0 d44 +
 0 d45 + 0 d46 + 0 d47 + 0 d48 + 0 d49 + 0 d50

 ..Constraints

 Arc 24: f1 + f2 + f3 - d24 <=  1200

 Arc 25: f4 + f5 + f6 - d25 <=  1200

 Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
         f14 + f15 + f16 + f17 + f18 + f19 + f20 +
         f21 + f22 + f23 + f24 - d26 <=  1200

 Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
         f32 + f33 - d27 <=  1200
```

```
Arc 28: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 - d28 <=  1200

Arc 29: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 - d29 <=  1200

Arc 30: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 - d30 <=  1200

Arc 31: f61 + f62 + f63 - d31 <=  1200

Arc 32: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 - d32 <=  1200

Arc 33: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 - d33 <=  1200

Arc 34: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 - d34 <=  4800

Arc 35: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 - d35 <=  4800

Arc 36: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 - d36 <=  4800

Arc 37: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 - d37 <=  4800

Arc 38: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 + f25 + f26 + f27 + f28 + f29 +
        f30 + f31 + f32 + f33 - d38 <=  4800

Arc 39: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 - d39 <=  4800

Arc 40: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 - d40 <=  4800

Arc 41: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 - d41 <=  4800

Arc 42: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 - d42 <=  4800

Arc 43: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
```

```
        f48 + f55 + f56 + f57 - d43 <=  4800

 Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 - d44 <=  4800

 Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - d45 <=  4800

 Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - d46 <=  4800

 Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - d47 <=  4800

 Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - d48 <=  4800

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - d49 <=  4800

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - d50 <=  4800

 Budget: 100 d24 + 100 d25 + 100 d26 + 100 d27 + 100 d28 + 100 d29 +
        100 d30 + 100 d31 + 100 d32 + 100 d33 + 100 d34 +
        100 d35 + 100 d36 + 100 d37 + 100 d38 + 100 d39 +
        100 d40 + 100 d41 + 100 d42 + 100 d43 + 100 d44 +
        100 d45 + 100 d46 + 100 d47 + 100 d48 + 100 d49 +
        100 d50  <= 100000

* ------ end ------ *



Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  398K bytes.
  Objective Function is MAXIMIZED.
  Variables:         90
  Constraints:       28
    28 LE,    0 EQ,    0 GE.
  Non-zero LP elements:      423
```

Disk Space: 0K bytes.
Page Space: 20K bytes.
Capacity: 12.1% used.
Estimated Time: 00:01:40

Iter 13
Solution Time: 00:00:03
A L T E R N A T E S O L U T I O N S

File: aimp11 11/25/88 18:47:09 Page 1-1
SOLUTION (Maximized): 237.6587 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
I f1	1,200.0000	0.0360	f2	0.0000	0.0216
f3	0.0000	0.0294	I f4	1,200.0000	0.0130
f5	0.0000	0.0078	f6	0.0000	0.0106
f7	0.0000	0.0002	f8	0.0000	0.0001
f9	0.0000	0.0002	f10	0.0000	0.0001
f11	0.0000	0.0001	f12	0.0000	0.0001
f13	0.0000	0.0002	f14	0.0000	0.0001
f15	0.0000	0.0001	f16	0.0000	0.0005
f17	0.0000	0.0003	f18	0.0000	0.0004
f19	0.0000	0.0003	f20	0.0000	0.0002

File: aimp11 11/25/88 18:47:09 Page 1-2
SOLUTION (Maximized): 237.6587 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
f21	0.0000	0.0003	f22	0.0000	0.0004
I f23	1,000.0000	0.0003	I f24	200.0000	0.0003
I f25	200.0000	0.0073	f26	0.0000	0.0044
I f27	1,000.0000	0.0059	f28	0.0000	0.0044

f29	0.0000	0.0026	f30	0.0000	0.0036
f31	0.0000	0.0059	f32	0.0000	0.0036
f33	0.0000	0.0048	f34	0.0000	0.0036
f35	0.0000	0.0022	f36	1,200.0000	0.0030
f37	0.0000	0.0022	f38	0.0000	0.0013
f39	0.0000	0.0018	f40	0.0000	0.0030

File: aimp11 11/25/88 18:47:09 Page 1-3
SOLUTION (Maximized): 237.6587 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	0.0018	f42	0.0000	0.0024
f43	0.0000	0.0022	f44	0.0000	0.0013
f45	1,200.0000	0.0018	f46	0.0000	0.0013
f47	0.0000	0.0008	f48	0.0000	0.0011
f49	0.0000	0.0018	f50	0.0000	0.0011
f51	0.0000	0.0015	f52	0.0000	0.0073
f53	0.0000	0.0044	f54	1,200.0000	0.0059
f55	0.0000	0.0044	f56	0.0000	0.0026
f57	0.0000	0.0036	f58	0.0000	0.0059
f59	0.0000	0.0036	f60	0.0000	0.0048

File: aimp11 11/25/88 18:47:09 Page 1-4
SOLUTION (Maximized): 237.6587 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
f61	2,200.0000	0.0720	f62	0.0000	0.0432
f63	0.0000	0.0588	d24	0.0000	0.0000
d25	0.0000	0.0000	d26	0.0000	0.0000
d27	0.0000	0.0000	d28	0.0000	0.0000
d29	0.0000	0.0000	d30	0.0000	0.0000

I	d31	1,000.0000	0.0000	d32	0.0000	0.0000	
	d33	0.0000	0.0000	d34	0.0000	0.0000	
	d35	0.0000	0.0000	d36	0.0000	0.0000	
	d37	0.0000	0.0000	d38	0.0000	0.0000	
	d39	0.0000	0.0000	d40	0.0000	0.0000	

SOLUTION (Maximized): 237.6587 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
d41	0.0000	0.0000	d42	0.0000	0.0000
d43	0.0000	0.0000	d44	0.0000	0.0000
d45	0.0000	0.0000	d46	0.0000	0.0000
d47	0.0000	0.0000	d48	0.0000	0.0000
d49	0.0000	0.0000	d50	0.0000	0.0000

CONSTRAINTS: Investment Strategy Model 1

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 24	1,200.0000 <	1,200.0000	Arc 25	1,200.0000 <	1,200.0000
Arc 26	1,200.0000 <	1,200.0000	Arc 27	1,200.0000 <	1,200.0000
Arc 28	1,200.0000 <	1,200.0000	Arc 29	1,200.0000 <	1,200.0000
Arc 30	1,200.0000 <	1,200.0000	Arc 31	1,200.0000 <	1,200.0000
Arc 32	0.0000 <	1,200.0000	Arc 33	1,200.0000 <	1,200.0000
Arc 34	1,200.0000 <	4,800.0000	Arc 35	1,200.0000 <	4,800.0000
Arc 36	1,200.0000 <	4,800.0000	Arc 37	0.0000 <	4,800.0000

I	Arc 38	2,400.0000 < 4,800.0000	Arc 39	4,800.0000 < 4,800.0000	
I	Arc 40	0.0000 < 4,800.0000	Arc 41	1,200.0000 < 4,800.0000	
I	Arc 42	4,800.0000 < 4,800.0000	Arc 43	0.0000 < 4,800.0000	

File: aimp11 11/25/88 18:47:09 Page 1-7

CONSTRAINTS: Investment Strategy Model 1

Constraint	Activity	RHS	Constraint	Activity	RHS
I Arc 44	1,200.0000 < 4,800.0000		I Arc 45	4,800.0000 < 4,800.0000	
I Arc 46	1,000.0000 < 4,800.0000		Arc 47	4,800.0000 < 4,800.0000	
Arc 48	4,800.0000 < 4,800.0000		I Arc 49	1,000.0000 < 4,800.0000	
I Arc 50	4,800.0000 < 4,800.0000		Budget	100000.0000 <100000.0000	

Total Error: 0.000000

D.8 Solution to Investment Strategy Model 1 (Case 2)

```
LP83 a:aimp12 output a:aimp12out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Investment Strategy Model 1

 ..Objective Maximize

 0.036 f1 + 0.0216 f2 + 0.0294 f3 + 0.01296 f4 +
 0.007776 f5 + 0.010584 f6 + 0.00021499 f7 + 0.00012899 f8 +
 0.00017558 f9 + 0.00012899 f10 + 0.0000774 f11 + 0.00010535 f12 +
 0.00017558 f13 + 0.00010535 f14 + 0.00014339 f15 + 0.00052255 f16 +
 0.00031353 f17 + 0.00042675 f18 + 0.00031353 f19 + 0.00018812 f20 +
 0.00025605 f21 + 0.00042675 f22 + 0.00025605 f23 + 0.00034851 f24 +
 0.0072576 f25 + 0.00435456 f26 + 0.00592704 f27 + 0.00435456 f28 +
 0.00261274 f29 + 0.00355622 f30 + 0.00592704 f31 + 0.00355622 f32 +
 0.00484042 f33 + 0.0036288 f34 + 0.00217728 f35 + 0.00296352 f36 +
 0.00217728 f37 + 0.00130637 f38 + 0.00177811 f39 + 0.00296352 f40 +
 0.00177811 f41 + 0.00242021 f42 + 0.00217728 f43 + 0.00130637 f44 +
 0.00177811 f45 + 0.00130637 f46 + 0.00078382 f47 + 0.00106687 f48 +
 0.00177811 f49 + 0.00106687 f50 + 0.00145212 f51 + 0.0072576 f52 +
 0.00435456 f53 + 0.00592704 f54 + 0.00435456 f55 + 0.00261274 f56 +
 0.00355622 f57 + 0.00592704 f58 + 0.00355622 f59 + 0.00484042 f60 +
 0.072 f61 + 0.0432 f62 + 0.0588 f63 +
 0 d24 + 0 d25 + 0 d26 + 0 d27 + 0 d28 + 0 d29 + 0 d30 +
 0 d31 + 0 d32 + 0 d33 + 0 d34 + 0 d35 + 0 d36 + 0 d37 +
 0 d38 + 0 d39 + 0 d40 + 0 d41 + 0 d42 + 0 d43 + 0 d44 +
 0 d45 + 0 d46 + 0 d47 + 0 d48 + 0 d49 + 0 d50

 ..Bounds

 d24 <= 100
 d25 <= 100
 d26 <= 100
 d27 <= 100
 d28 <= 100
 d29 <= 100
 d30 <= 100
 d31 <= 100
 d32 <= 100
 d33 <= 100
 d34 <= 100
```

```
d35 <= 100
d36 <= 100
d37 <= 100
d38 <= 100
d39 <= 100
d40 <= 100
d41 <= 100
d42 <= 100
d43 <= 100
d44 <= 100
d45 <= 100
d46 <= 100
d47 <= 100
d48 <= 100
d49 <= 100
d50 <= 100


..Constraints

Arc 24: f1 + f2 + f3 - d24 <=  1200

Arc 25: f4 + f5 + f6 - d25 <=  1200

Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 + f16 + f17 + f18 + f19 + f20 +
        f21 + f22 + f23 + f24 - d26 <=  1200

Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
        f32 + f33 - d27 <=  1200

Arc 28: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 - d28 <=  1200

Arc 29: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 - d29 <=  1200

Arc 30: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 - d30 <=  1200

Arc 31: f61 + f62 + f63 - d31 <=  1200

Arc 32: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 - d32 <=  1200

Arc 33: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 - d33 <=  1200

Arc 34: f34 + f35 + f36 + f37 + f38 + f39 + f40 +
        f41 + f42 - d34 <=  4800
```

```
Arc 35: f43 + f44 + f45 + f46 + f47 + f48 + f49 +
        f50 + f51 - d35 <=  4800

Arc 36: f52 + f53 + f54 + f55 + f56 + f57 + f58 +
        f59 + f60 - d36 <=  4800

Arc 37: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
        f14 + f15 - d37 <=  4800

Arc 38: f16 + f17 + f18 + f19 + f20 + f21 + f22 +
        f23 + f24 + f25 + f26 + f27 + f28 + f29 +
        f30 + f31 + f32 + f33 - d38 <=  4800

Arc 39: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 - d39 <=  4800

Arc 40: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 - d40 <=  4800

Arc 41: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 - d41 <=  4800

Arc 42: f7 + f8 + f9 + f16 + f17 + f18 + f25 +
        f26 + f27 + f34 + f35 + f36 + f43 + f44 +
        f45 + f52 + f53 + f54 - d42 <=  4800

Arc 43: f10 + f11 + f12 + f19 + f20 + f21 + f28 +
        f29 + f30 + f37 + f38 + f39 + f46 + f47 +
        f48 + f55 + f56 + f57 - d43 <=  4800

Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 - d44 <=  4800

Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - d45 <=  4800

Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - d46 <=  4800

Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - d47 <=  4800

Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
```

```
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - d48 <=  4800

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - d49 <=  4800

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - d50 <=  4800

 Budget: 100 d24 + 100 d25 + 100 d26 + 100 d27 + 100 d28 + 100 d29 +
        100 d30 + 100 d31 + 100 d32 + 100 d33 + 100 d34 +
        100 d35 + 100 d36 + 100 d37 + 100 d38 + 100 d39 +
        100 d40 + 100 d41 + 100 d42 + 100 d43 + 100 d44 +
        100 d45 + 100 d46 + 100 d47 + 100 d48 + 100 d49 +
        100 d50  <= 100000

* ------ end ------ *



Statistics-
  LP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  398K bytes.
  Objective Function is MAXIMIZED.
  Variables:        90
  Constraints:      28
    28 LE,   0 EQ,   0 GE.
  Non-zero LP elements:     423
  Disk Space:     0K bytes.
  Page Space:    20K bytes.
  Capacity: 12.1% used.
  Estimated Time: 00:01:40

Iter   25
Solution Time:  00:00:04
A L T E R N A T E   S O L U T I O N S



File: aimp12                                        11/25/88 18:48:05 Page 1-1
SOLUTION (Maximized):    180.4016   Investment Strategy Model 1
-------------------------------------------------------------------------------
| Variable |  Activity   |    Cost     | Variable |  Activity   |    Cost     |
-------------------------------------------------------------------------------
I       f1    1,300.0000        0.0360 |       f2        0.0000        0.0216 |
-------------------------------------------------------------------------------
|       f3        0.0000        0.0294 I       f4    1,300.0000        0.0130 |
```

f5	0.0000	0.0078	f6	0.0000	0.0106
f7	0.0000	0.0002	f8	0.0000	0.0001
f9	0.0000	0.0002	f10	0.0000	0.0001
f11	0.0000	0.0001	f12	0.0000	0.0001
f13	0.0000	0.0002	f14	0.0000	0.0001
f15	0.0000	0.0001	f16	0.0000	0.0005
f17	0.0000	0.0003	f18	0.0000	0.0004
f19	0.0000	0.0003	f20	0.0000	0.0002

SOLUTION (Maximized): 180.4016 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
f21	0.0000	0.0003	f22	0.0000	0.0004
f23	600.0000	0.0003	f24	600.0000	0.0003
f25	1,000.0000	0.0073	f26	0.0000	0.0044
f27	300.0000	0.0059	f28	0.0000	0.0044
f29	0.0000	0.0026	f30	0.0000	0.0036
f31	0.0000	0.0059	f32	0.0000	0.0036
f33	0.0000	0.0048	f34	0.0000	0.0036
f35	0.0000	0.0022	f36	1,300.0000	0.0030
f37	0.0000	0.0022	f38	0.0000	0.0013
f39	0.0000	0.0018	f40	0.0000	0.0030

SOLUTION (Maximized): 180.4016 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	0.0018	f42	0.0000	0.0024
f43	0.0000	0.0022	f44	0.0000	0.0013

I	f45	950.0000	0.0018	f46	0.0000	0.0013	
	f47	0.0000	0.0008	f48	0.0000	0.0011	
	f49	0.0000	0.0018	f50	0.0000	0.0011	
I	f51	350.0000	0.0015	f52	0.0000	0.0073	
	f53	0.0000	0.0044	I f54	1,300.0000	0.0059	
	f55	0.0000	0.0044	f56	0.0000	0.0026	
	f57	0.0000	0.0036	f58	0.0000	0.0059	
	f59	0.0000	0.0036	f60	0.0000	0.0048	

File: aimp12 11/25/88 18:48:05 Page 1-4
SOLUTION (Maximized): 180.4016 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
I f61	1,300.0000	0.0720	f62	0.0000	0.0432
f63	0.0000	0.0588	d24	100.0000	0.0000
d25	100.0000	0.0000	d26	0.0000	0.0000
d27	100.0000	0.0000	d28	100.0000	0.0000
d29	100.0000	0.0000	d30	100.0000	0.0000
d31	100.0000	0.0000	d32	0.0000	0.0000
d33	0.0000	0.0000	d34	0.0000	0.0000
d35	0.0000	0.0000	d36	0.0000	0.0000
d37	0.0000	0.0000	d38	0.0000	0.0000
I d39	50.0000	0.0000	d40	0.0000	0.0000

File: aimp12 11/25/88 18:48:05 Page 1-5
SOLUTION (Maximized): 180.4016 Investment Strategy Model 1

Variable	Activity	Cost	Variable	Activity	Cost
d41	0.0000	0.0000	I d42	50.0000	0.0000
d43	0.0000	0.0000	d44	0.0000	0.0000
I d45	100.0000	0.0000	d46	0.0000	0.0000

D.9 Solution to Investment Strategy Model 2 (Case 3)

```
MIP83 a:aimp23 output a:aimp23out alternate 1

  Copyright (C) 1986 by Sunset Software.
  All Rights Reserved Worldwide.
  1613 Chelsea Road,  Suite 153
  San Marino, California  91108   U.S.A.

  Licensed Solely To: Wright-Patterson Air Force Base

 ..Title

 Investment Strategy Model 2

 ..Objective Maximize

 0.036 f1 + 0.0216 f2 + 0.0294 f3 + 0.01296 f4 +
 0.007776 f5 + 0.010584 f6 + 0.00021499 f7 + 0.00012899 f8 +
 0.00017558 f9 + 0.00012899 f10 + 0.0000774 f11 + 0.00010535 f12 +
 0.00017558 f13 + 0.00010535 f14 + 0.00014339 f15 + 0.00052255 f16 +
 0.00031353 f17 + 0.00042675 f18 + 0.00031353 f19 + 0.00018812 f20 +
 0.00025605 f21 + 0.00042675 f22 + 0.00025605 f23 + 0.00034851 f24 +
 0.0072576 f25 + 0.00435456 f26 + 0.00592704 f27 + 0.00435456 f28 +
 0.00261274 f29 + 0.00355622 f30 + 0.00592704 f31 + 0.00355622 f32 +
 0.00484042 f33 + 0.0036288 f34 + 0.00217728 f35 + 0.00296352 f36 +
 0.00217728 f37 + 0.00130637 f38 + 0.00177811 f39 + 0.00296352 f40 +
 0.00177811 f41 + 0.00242021 f42 + 0.00217728 f43 + 0.00130637 f44 +
 0.00177811 f45 + 0.00130637 f46 + 0.00078382 f47 + 0.00106687 f48 +
 0.00177811 f49 + 0.00106687 f50 + 0.00145212 f51 + 0.0072576 f52 +
 0.00435456 f53 + 0.00592704 f54 + 0.00435456 f55 + 0.00261274 f56 +
 0.00355622 f57 + 0.00592704 f58 + 0.00355622 f59 + 0.00484042 f60 +
 0.072 f61 + 0.0432 f62 + 0.0588 f63 +
 [  0 g24 + 0 g25 + 0 g26 + 0 g27 + 0 g28 + 0 g29 + 0 g30 +
 0 g31 + 0 g32 + 0 g33 + 0 g34 + 0 g35 + 0 g36 + 0 g37 +
 0 g38 + 0 g39 + 0 g40 + 0 g41 + 0 g42 + 0 g43 + 0 g44 +
 0 g45 + 0 g46 + 0 g47 + 0 g48 + 0 g49 + 0 g50 ]

 ..Constraints

 Arc 24: f1 + f2 + f3 - 100 g24 <=  1200

 Arc 25: f4 + f5 + f6 - 100 g25 <=  1200

 Arc 26: f7 + f8 + f9 + f10 + f11 + f12 + f13 +
         f14 + f15 + f16 + f17 + f18 + f19 + f20 +
         f21 + f22 + f23 + f24 - 100 g26 <=  1200

 Arc 27: f25 + f26 + f27 + f28 + f29 + f30 + f31 +
         f32 + f33 - 100 g27 <=  1200
```

```
        f48 + f55 + f56 + f57 - 100 g43 <=  4800

 Arc 44: f13 + f14 + f15 + f22 + f23 + f24 + f31 +
        f32 + f33 + f40 + f41 + f42 + f49 + f50 +
        f51 + f58 + f59 + f60 - 100 g44 <=  4800

 Arc 45: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - 100 g45 <=  4800

 Arc 46: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - 100 g46 <=  4800

 Arc 47: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - 100 g47 <=  4800

 Arc 48: f1 + f4 + f7 + f10 + f13 + f16 + f19 +
        f22 + f25 + f28 + f31 + f34 + f37 + f40 +
        f43 + f46 + f49 + f52 + f55 + f58 + f61 - 100 g48 <=  4800

 Arc 49: f2 + f5 + f8 + f11 + f14 + f17 + f20 +
        f23 + f26 + f29 + f32 + f35 + f38 + f41 +
        f44 + f47 + f50 + f53 + f56 + f59 + f62 - 100 g49 <=  4800

 Arc 50: f3 + f6 + f9 + f12 + f15 + f18 + f21 +
        f24 + f27 + f30 + f33 + f36 + f39 + f42 +
        f45 + f48 + f51 + f54 + f57 + f60 + f63 - 100 g50 <=  4800

 Budget: 10000 g24 + 10000 g25 + 10000 g26 + 10000 g27 + 10000 g28 +
        10000 g29 + 10000 g30 + 10000 g31 + 10000 g32 + 10000 g33 +
        10000 g34 + 10000 g35 + 10000 g36 + 10000 g37 + 10000 g38 +
        10000 g39 + 10000 g40 + 10000 g41 + 10000 g42 + 10000 g43 +
        10000 g44 + 10000 g45 + 10000 g46 + 10000 g47 + 10000 g48 +
        10000 g49 + 10000 g50  <= 100000

* ------ end ------ *



Statistics-
  MIP83 Version 5.00a
  Machine memory:  640K bytes.
  Pagable memory:  280K bytes.
  Objective Function is MAXIMIZED.
  MIP Strategy:      1
  Variables:        90
    Integer:        27
  Constraints:      28
```

```
   28 LE,   0 EQ,   0 GE.
  Non-zero LP elements:     423
  Disk Space:      0K bytes.
  Page Space:     20K bytes.
  Capacity: 12.1% used.
  Estimated Time: 00:01:40

Iter   14
Solution Time:  00:00:03
A L T E R N A T E   S O L U T I O N S

INTEGER SOLUTION
```

File: aimp23 11/25/88 19:02:30 Page 1-1
SOLUTION (Maximized): 237.6587 Investment Strategy Model 2

	Variable	Activity	Cost		Variable	Activity	Cost
I	f1	1,200.0000	0.0360		f2	0.0000	0.0216
	f3	0.0000	0.0294	I	f4	1,200.0000	0.0130
	f5	0.0000	0.0078		f6	0.0000	0.0106
	f7	0.0000	0.0002		f8	0.0000	0.0001
	f9	0.0000	0.0002		f10	0.0000	0.0001
	f11	0.0000	0.0001		f12	0.0000	0.0001
	f13	0.0000	0.0002	I	f14	0.0000	0.0001
	f15	0.0000	0.0001		f16	0.0000	0.0005
	f17	0.0000	0.0003		f18	0.0000	0.0004
	f19	0.0000	0.0003		f20	0.0000	0.0002

File: aimp23 11/25/88 19:02:30 Page 1-2
SOLUTION (Maximized): 237.6587 Investment Strategy Model 2

	Variable	Activity	Cost		Variable	Activity	Cost
	f21	0.0000	0.0003		f22	0.0000	0.0004
I	f23	1,000.0000	0.0003	I	f24	200.0000	0.0003
I	f25	200.0000	0.0073		f26	0.0000	0.0044

I	f27	1,000.0000	0.0059	f28	0.0000	0.0044	
	f29	0.0000	0.0026	f30	0.0000	0.0036	
	f31	0.0000	0.0059	f32	0.0000	0.0036	
	f33	0.0000	0.0048	f34	0.0000	0.0036	
	f35	0.0000	0.0022	I f36	1,200.0000	0.0030	
	f37	0.0000	0.0022	f38	0.0000	0.0013	
	f39	0.0000	0.0018	f40	0.0000	0.0030	

File: aimp23 11/25/88 19:02:30 Page 1-3
SOLUTION (Maximized): 237.6587 Investment Strategy Model 2

Variable	Activity	Cost	Variable	Activity	Cost
f41	0.0000	0.0018	f42	0.0000	0.0024
f43	0.0000	0.0022	f44	0.0000	0.0013
I f45	1,200.0000	0.0018	f46	0.0000	0.0013
f47	0.0000	0.0008	f48	0.0000	0.0011
f49	0.0000	0.0018	f50	0.0000	0.0011
I f51	0.0000	0.0015	f52	0.0000	0.0073
f53	0.0000	0.0044	I f54	1,200.0000	0.0059
f55	0.0000	0.0044	f56	0.0000	0.0026
f57	0.0000	0.0036	f58	0.0000	0.0059
f59	0.0000	0.0036	f60	0.0000	0.0048

File: aimp23 11/25/88 19:02:30 Page 1-4
SOLUTION (Maximized): 237.6587 Investment Strategy Model 2

Variable	Activity	Cost	Variable	Activity	Cost
I f61	2,200.0000	0.0720	f62	0.0000	0.0432
f63	0.0000	0.0588	g24	0.0000	0.0000
g25	0.0000	0.0000	g26	0.0000	0.0000

	g27	0.0000	0.0000		g28	0.0000	0.0000
	g29	0.0000	0.0000		g30	0.0000	0.0000
I	g31	10.0000	0.0000		g32	0.0000	0.0000
	g33	0.0000	0.0000		g34	0.0000	0.0000
	g35	0.0000	0.0000		g36	0.0000	0.0000
	g37	0.0000	0.0000		g38	0.0000	0.0000
	g39	0.0000	0.0000		g40	0.0000	0.0000

File: aimp23 11/25/88 19:02:30 Page 1-5
SOLUTION (Maximized): 237.6587 Investment Strategy Model 2

Variable	Activity	Cost	Variable	Activity	Cost
g41	0.0000	0.0000	g42	0.0000	0.0000
g43	0.0000	0.0000	g44	0.0000	0.0000
g45	0.0000	0.0000	g46	0.0000	0.0000
g47	0.0000	0.0000	g48	0.0000	0.0000
g49	0.0000	0.0000	g50	0.0000	0.0000

File: aimp23 11/25/88 19:02:30 Page 1-6
CONSTRAINTS: Investment Strategy Model 2

Constraint	Activity	RHS	Constraint	Activity	RHS
Arc 24	1,200.0000 <	1,200.0000	Arc 25	1,200.0000 <	1,200.0000
Arc 26	1,200.0000 <	1,200.0000	Arc 27	1,200.0000 <	1,200.0000
Arc 28	1,200.0000 <	1,200.0000	Arc 29	1,200.0000 <	1,200.0000
Arc 30	1,200.0000 <	1,200.0000	Arc 31	1,200.0000 <	1,200.0000
Arc 32	0.0000 <	1,200.0000	Arc 33	1,200.0000 <	1,200.0000
Arc 34	1,200.0000 <	4,800.0000	Arc 35	1,200.0000 <	4,800.0000

I	Arc 36	1,200.0000 < 4,800.0000	I	Arc 37	0.0000 < 4,800.0000
I	Arc 38	2,400.0000 < 4,800.0000	I	Arc 39	4,800.0000 < 4,800.0000
I	Arc 40	0.0000 < 4,800.0000	I	Arc 41	1,200.0000 < 4,800.0000
I	Arc 42	4,800.0000 < 4,800.0000	I	Arc 43	0.0000 < 4,800.0000

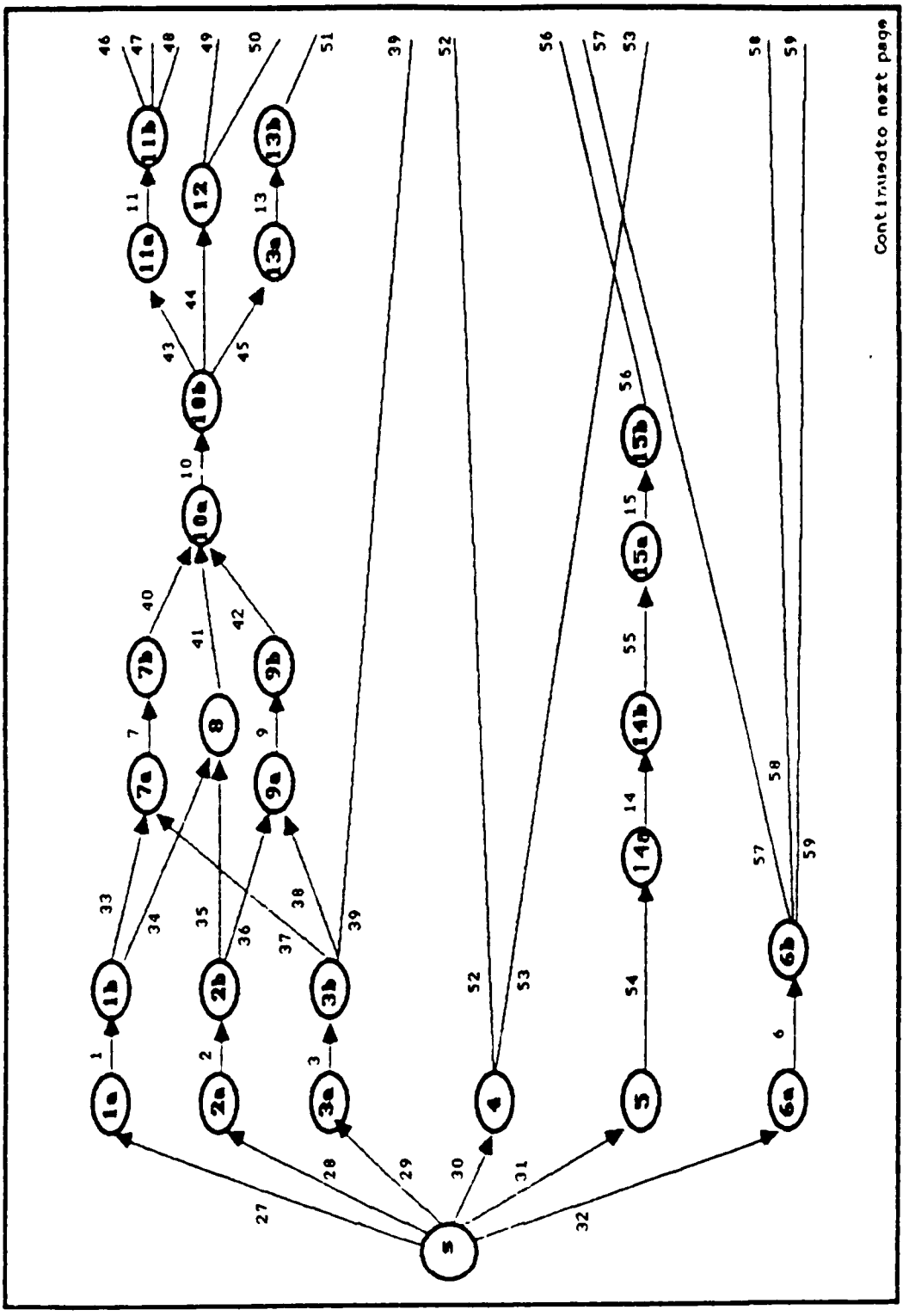
File: aimp23 11/25/88 19:02:30 Page 1-7

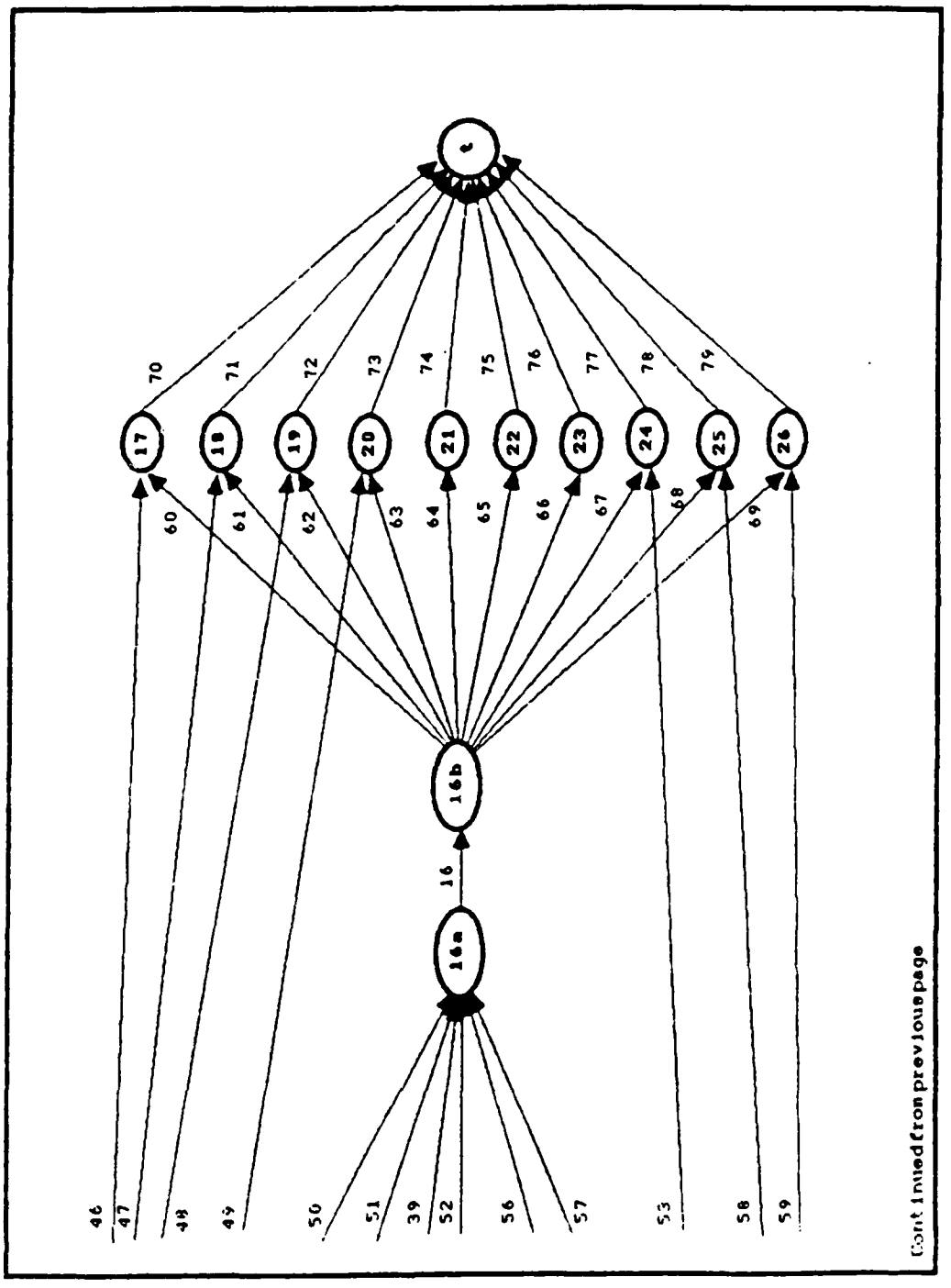
CONSTRAINTS: Investment Strategy Model 2

Constraint	Activity	RHS	Constraint	Activity	RHS
I Arc 44	1,200.0000 < 4,800.0000	I	Arc 45	4,800.0000 < 4,800.0000	
I Arc 46	1,000.0000 < 4,800.0000	I	Arc 47	4,800.0000 < 4,800.0000	
I Arc 48	4,800.0000 < 4,800.0000	I	Arc 49	1,000.0000 < 4,800.0000	
I Arc 50	4,800.0000 < 4,800.0000	I	Budget	100000.0000 <100000.0000	

Total Error: 0.000000

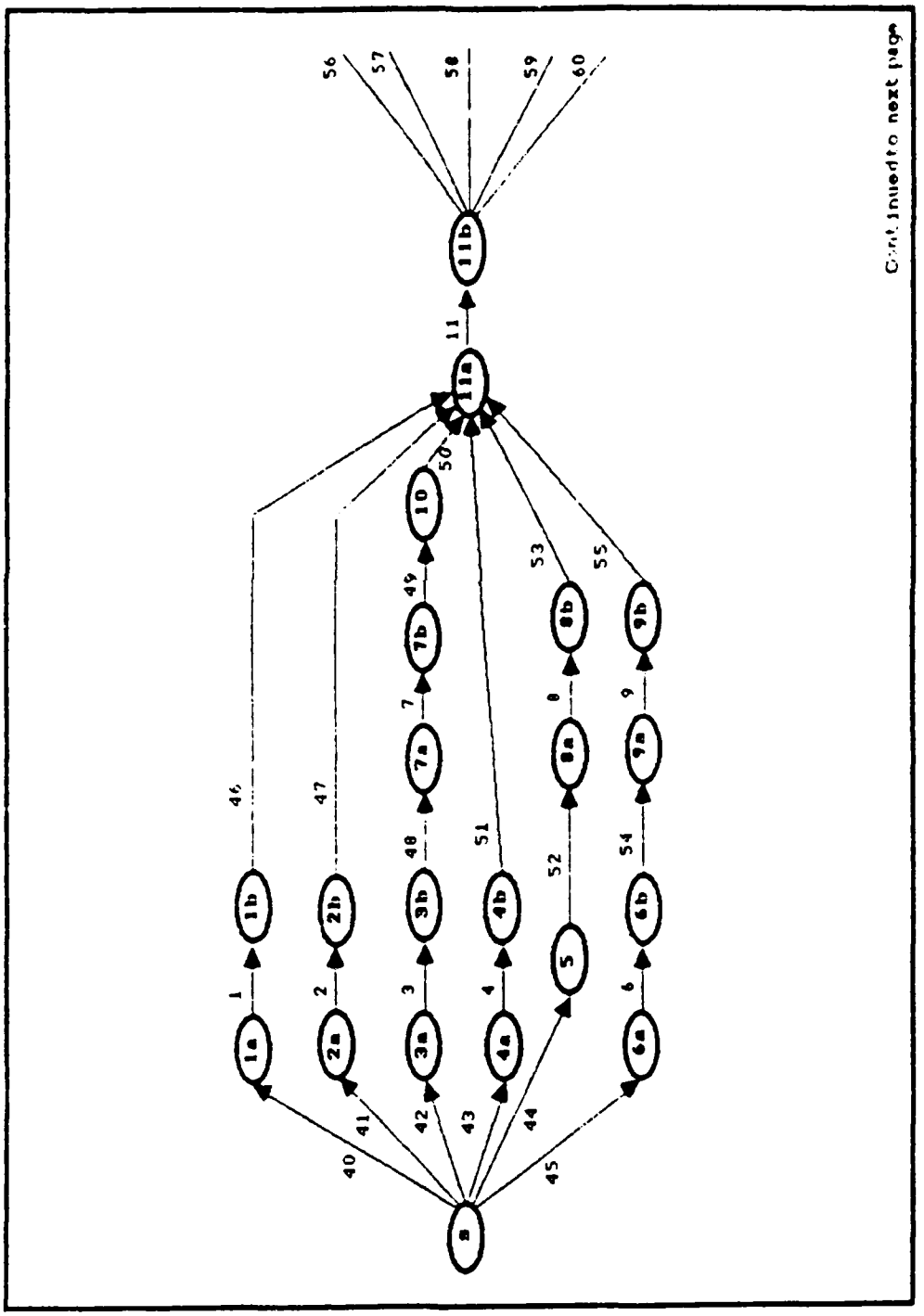
Appendix E. *Revised Network B*



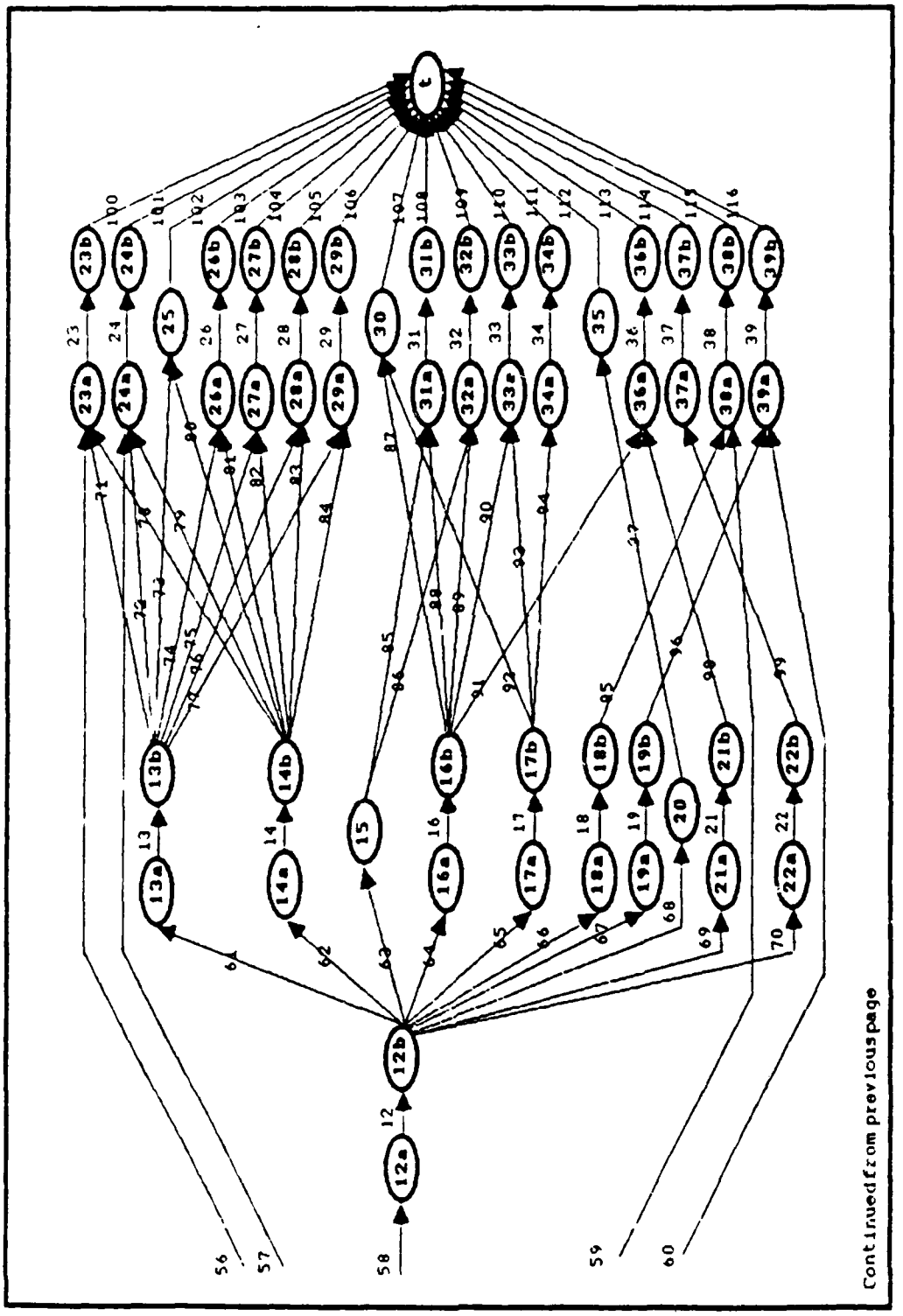


Revised Topology Of Network B

Appendix F. *Revised Network C*



Revised Topology Of Network C



Revised Topology Of Network C

Vita

Captain Eugene Yim was born on [redacted] He graduated [redacted] High School [redacted] He attended the University of Texas at Austin from which he received the degree of Bachelor of Science in Electrical Engineering in May, 1984. Upon graduation, he entered USAF Officer Training School and was commissioned in August, 1984. Captain Yim's first assignment was a three year tour as a Deputy Chief of Munitions Test at Hill AFB, Utah. While assigned at Hill, he received a Master of Engineering Administration degree from the University of Utah. He entered the School of Engineering, Air Force Institute of Technology, in May, 1987.

[redacted]

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION: Unclassified	1b. RESTRICTIVE MARKINGS: None
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): AFIT/GOR/ENS/88D-25	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: School of Engineering; 6b. OFFICE SYMBOL (If applicable): AFIT/ENS	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code): Air Force Institute of Technology, Wright-Patterson AFB, Ohio 45433	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION: Department of Defense; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10. SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO.

11. TITLE (Include Security Classification)

Improving the Survivability of a Stochastic Communication Network (U)

12. PERSONAL AUTHOR(S)

Yim, Eugene, Capt, USAF

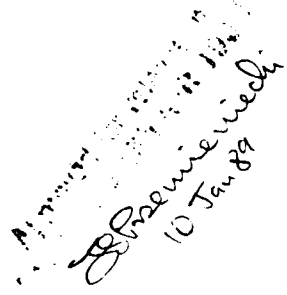
13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15. PAGE COUNT
MS Thesis	FROM _____ TO _____	1988, December	206

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	03		Stochastic Network, Communication Network, Maximum Flow, Expected Maximum Flow, Survival Probability,(See Reverse)
12	04		

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

See Reverse



20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: ☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☐ DTIC USERS	21. ABSTRACT SECURITY CLASSIFICATION: Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL: Dr. Yupo, Chan	22b. TELEPHONE (Include Area Code): (513) 255-3362 / 22c. OFFICE SYMBOL: AFIT/ENS

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE

Unclassified